UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Matching Biomedical Knowledge Graphs with Neural Embeddings

Rodrigo David Rodrigues Neves

Mestrado em Ciência de Dados

Dissertação orientada por:

Professora Doutora Cátia Luísa Santana Calisto Pesquita

2020

"You do not rise to the level of your goals. You fall to the level of your systems."

— James Clear, *Atomic Habits*

Acknowledgements

First of all, I must thank Professor Cátia Pesquita for her unswerving support and enlightenment. Her astuteness, dedication, patience, humour and creativity were essential for me to develop my best work. A special thanks to Daniel Faria for his technical expertise.

I would also like to express my gratitude to my parents and grandparents for their unconditional love and support, and for always being there in the many ups and downs.

I extend my thankfulness to all my LASIGE fellow colleagues, Little Big Apple dancing partners and close friends for every listening ear and word of encouragement.

Finally, I would like to thank Fundação para a Ciência e a Tecnologia, which provided the funding through LASIGE (UID/CEC/00408/2019) and by SMILAX Project (PTDC/EEI-ESS/4633/2014; UIDB/00408/2020).

Resumo

Os grafos de conhecimento são estruturas que se tornaram fundamentais para a organização dos dados biomédicos que têm sido produzidos a um ritmo exponencial nos últimos anos. A abrangente adoção desta forma de estruturar e descrever dados levou ao desenvolvimento de abordagens de prospeção de dados que tirassem partido desta informação com o intuito de auxiliar o progresso do conhecimento científico. Porém, devido à impossibilidade de isolamento de domínios de conhecimento e à idiossincrasia humana, grafos de conhecimento construídos por diferentes indivíduos contêm muitas vezes conceitos equivalentes descritos de forma diferente, dificultando uma análise integrada de dados de diferentes grafos de conhecimento. Vários sistemas de alinhamento de grafos de conhecimento têm focado a resolução deste desafio. Contudo, o desempenho destes sistemas no alinhamento de grafos de conhecimento biomédicos estagnou nos últimos quatro anos com algoritmos e recursos externos bastante trabalhados para aprimorar os resultados.

Nesta dissertação, apresentamos duas novas abordagens de alinhamento de grafos de conhecimento empregando *Neural Embeddings*: uma utilizando semelhança simples entre *embeddings* à base de palavras e de entidades de grafos; outra treinando um modelo mais complexo que refinasse a informação proveniente de *embeddings* baseados em palavras. A metodologia proposta visa integrar estas abordagens no processo regular de alinhamento, utilizando como infraestrutura o sistema *AgreementMakerLight*. Estas novas componentes permitem extender os algoritmos de alinhamento do sistema, descobrindo novos mapeamentos, e criar uma abordagem de alinhamento mais generalizável e menos dependente de ontologias biomédicas externas.

Esta nova metodologia foi avaliada em três casos de teste de alinhamento de ontologias biomédicas, provenientes da *Ontology Alignment Evaluation Initiative*. Os resultados demonstraram que apesar de ambas as abordagens não excederem o estado da arte, estas obtiveram um desempenho benéfico nas tarefas de alinhamento, superando a performance de todos os sistemas que não usam ontologias externas e inclusive alguns que tiram proveito das mesmas, o que demonstra o valor das técnicas de *Neural Embeddings* na tarefa de alinhamento de grafos do conhecimento biomédicos.

Palavras Chave: alinhamento de grafos de conhecimento, representações vetoriais neuronais, ontologias biomédicas.

Abstract

Knowledge graphs are data structures which became essential to organize biomedical data produced at an exponential rate in the last few years. The broad adoption of this method of structuring and describing data resulted in the increased interest to develop data mining approaches which took advantage of these information structures in order to improve scientific knowledge. However, due to human idiosyncrasy and also the impossibility to isolate knowledge domains in separate pieces, knowledge graphs constructed by different individuals often contain equivalent concepts described differently. This obstructs the path to an integrated analysis of data described by multiple knowledge graphs. Multiple knowledge graph matching systems have been developed to address this challenge. Nevertheless, the performance of these systems has stagnated in the last four years, despite the fact that they were provided with highly tailored algorithms and external resources to tackle this task.

In this dissertation, we present two novel knowledge graph matching approaches employing neural embeddings: one using plain embedding similarity based on word and graph models; the other one using a more complex word-based model which requires training data to refine embeddings. The proposed methodology aims to integrate these approaches in the regular matching process, using the AgreementMakerLight system as a foundation. These new components enable the extension of the system's current matching algorithms, discovering new mappings, and developing a more generalizable and less dependent on external biomedical ontologies matching procedure.

This new methodology was evaluated on three biomedical ontology matching test cases provided by the Ontology Alignment Evaluation Initiative. The results showed that despite both embedding approaches don't exceed state of the art results, they still produce better results than any other matching systems which do not make use of external ontologies and also surpass some that do benefit from them. This shows that Neural Embeddings are a valuable technique to tackle the challenge of biomedical knowledge graph matching.

Keywords: knowledge graph matching, neural embeddings, biomedical ontologies.

Resumo Alargado

Nas últimas décadas, a produção de dados biomédicos tem expandido a um ritmo exponencial. Para além do aumento significativo do volume de dados, estes evoluíram em termos de complexidade e heterogeneidade, devido ao desenvolvimento científico e tecnológico. Esta transformação levou à concepção de novas estruturas que permitissem armazenar esta informação de forma interligada, padronizada, descritiva e acessível, de modo a ser interpretável tanto por humanos como por máquinas, os grafos de conhecimento.

Os grafos de conhecimento são estruturas em formato de grafo que descrevem características de entidades reais e relações entre si, através de ligações a conceitos descritos em ontologias. Uma ontologia é um documento formal que representa detalhadamente conceitos e respetivas inter-relações, referentes a um determinado domínio. Estes documentos permitem ter uma descrição semântica pormenorizada e padronizada, e delinear restrições lógicas relacionadas com os conceitos descritos. Esta abordagem de representação de conhecimento foi largamente adotada em vários domínios, tendo uma notável importância no ramo das Ciências da Vida e no domínio biomédico.

Contudo, esta representação de conhecimento expõe um desafio à análise integrada de dados descritos em múltiplos grafos, a existência de conceitos repetidos em diferentes grafos sem quaisquer ligações entre si. Devido ao obstáculo da idiossincrasia humana, grafos concebidos por diferentes indivíduos certamente terão perspetivas e vocabulários distintos. Para além disso, a natureza interligada do mundo que nos rodeia impossibilita a separação de conceitos por domínios isolados. Por outro lado, a procura e utilização de técnicas de prospeção de dados combinadas com grafos de conhecimento tem crescido pela sua capacidade de encontrar padrões profundos nos dados e auxiliar a evolução do conhecimento científico.

Por todos esses motivos, a fim de possibilitar a integração de dados de diferentes grafos de conhecimento, é necessário identificar as interseções entre eles, mapeando ligações entre entidades equivalentes. Este processo tem o nome de alinhamento de grafos de conhecimento e pode ser subdivido em duas tarefas: alinhamento de ontologias e alinhamento de instâncias. Como a denominação indica, o alinhamento de ontologias passa por descobrir conceitos equivalentes entre duas ontologias, e o alinhamento de instâncias passa por encontrar instâncias/entidades que sejam iguais.

É impraticável executar este processo manualmente. O alinhamento de duas ontologias com menos de 60,000 conceitos cada demorou cerca de sete *person-years*. Algumas ontologias biomédicas contêm mais de 300,000 conceitos. Para além da complexidade colossal de avaliar biliões de combinações de conceitos, apenas peritos do próprio domínio estão aptos para discernir as relações entre diferentes conceitos.

Dada esta adversidade, têm sido desenvolvidos programas computacionais com a finalidade de automatizar a tarefa de alinhamento de ontologias. Desde 2004, o projeto *Ontology Alignment Evaluation Initiative* (OAEI) tem avaliado anualmente a performance destes sistemas em ontologias de teste que incluem em parte ontologias biomédicas de tamanho considerável. Apesar de a maioria dos sistemas participantes ter progredido gradualmente ao longo do tempo, nos últimos quatro anos verificou-se uma estagnação de performance dos sistemas com melhores classificações nas ontologias do domínio biomédico. Isto ocorre pelo facto de o desenvolvimento de estratégias de alinhamento para ontologias biomédicas ser uma tarefa complexa, devido à abundância de classes que aumentam a complexidade de combinações, vocabulário refinado e heterogéneo, e diversos sinónimos por conceito que dificultam a percepção de que nomes são mais relevantes na representação de um conceito. Por outro lado, os algoritmos de alinhamento de ontologias são bastante trabalhados e adaptados para obterem uma boa performance nas ontologias de teste, usando recursos como ontologias externas com informação relacionada com as ontologias a alinhar como conhecimento complementar, o que não permite uma utilização geral destes sistemas. Assim sendo, nesta dissertação apresentamos uma nova abordagem de alinhamento de ontologias empregando *Neural Embeddings*.

Neural Embeddings é uma técnica de aprendizagem automática que transforma variáveis categóricas em vetores multidimensionais de valores reais. Esta técnica treina um modelo através de uma rede neuronal e produz um espaço vetorial de baixa dimensão, onde estão representadas as variáveis apresentadas na fase de treino. O que torna este modelo significante é o facto de variáveis semelhantes serem representadas por vetores próximos entre si. Esta técnica tem sido aplicada em várias tarefas de processamento de linguagem natural, através de modelos de representação de palavras, e para representação de entidades em grafos. A hipótese que orientou esta dissertação foi que a técnica *neural embeddings* poderia ser benéfica para a tarefa de alinhamento de ontologias através de duas componentes: modelos de palavras, providenciando informação semântica que poderia estar em falta nas ontologias; modelos de entidades de grafos, interpretando a estrutura dos grafos das ontologias que poderiam não ser tão facilmente compreendidas através técnicas de alinhamento convencionais.

A metodologia proposta visa criar uma abordagem de alinhamento mais generalizável e passa por integrar uma fase de cálculo de semelhança de *embeddings* no processo de alinhamento de ontologias. Para tal, tirámos proveito do sistema de alinhamento de ontologias *AgreementMakerLight* como infraestrutura para incluir as nossas estratégias de alinhamento à base de *embeddings*. As estratégias implementadas baseiam-se em dois tipos de abordagens: a primeira, utilizando simples semelhanças entre vetores, provenientes de modelos palavras e de modelos de grafos, permitindo uma aplicação mais generalizável; a segunda, utilizando um modelo mais complexo que refina a informação existente nos modelos de *em-*

beddings para focar a tarefa de alinhamento, para analisar quão divergente é o desempenho entre uma alternativa mais simples contra outra mais complexa e direcionada. Na primeira abordagem foram testados modelos de palavras e modelos de grafos isolados e posteriormente combinados entre si e também com um algoritmo de semelhança entre sequências de caracteres. Na segunda abordagem foi experimentado um modelo de palavras aprimorado para representar frases em vez de palavras únicas e com uma ferramenta para excluir falsos positivos cuja semelhança de *embeddings* não representasse semelhança semântica.

Esta metodologia foi implementada de forma exploratória e iterativa, e avaliada em três casos de teste disponibilizados na edição de 2019 da OAEI, referentes a ontologias biomédicas de pequena e média dimensão, e comparada com os sistemas participantes daquele ano. O desempenho das abordagens foi avaliado de acordo com os alinhamentos de referência fornecidos pela iniciativa, utilizando as métricas precisão, *recall* e *F-Measure*.

Os resultados demonstraram que tanto a abordagem de semelhanças simples como do modelo complexo conseguem acrescentar valor a um algoritmo de alinhamento simples, sendo o ganho geralmente superior na abordagem mais complexa. Contudo, num dos casos de teste a abordagem mais simples superou a mais complexa, o que indica que nem sempre é a alternativa mais desejada. Nas tabelas de classificação dos sistemas participantes da OAEI, ambas as abordagens alcançaram sempre lugares na primeira metade das tabelas, superando todos os sistemas que não beneficiam de informação de ontologias externas e inclusive alguns dos sistemas que tiram proveito da mesma. Não obstante, nenhuma das abordagens conseguiu melhorar o estado da arte. Os modelos de palavras apresentam uma semelhança pouco profunda entre palavras, em vez de fortemente focada na semântica. Por outro lado, os modelos de grafos escolhidos são bastante dispendiosos em termos computacionais e de memória, tanto no período de treino como na fase de alinhamento, impossibilitando a sua aplicação em casos de teste de maior dimensão.

Dados estes desafios, como trabalho futuro tencionamos desenvolver algoritmos que calculem *embeddings* baseados em grafos de forma mais eficiente. Pretendemos também refinar os modelos baseados em palavras e treinar outros tipos de modelos como modelos baseados em frases que possam representar as entidades biomédicas com maior profundidade. Para além disso, tencionamos também estender a avaliação das abordagens desenvolvidas para grafos de conhecimento noutros domínios.

Contents

List of Figures

List of Tables

Acronyms

AML AgreementMakerLight.

API Application Programming Interface.

CAT Chinese Agricultural Thesaurus.

CBOW Continuous Bag of Words.

CPU Central Processing Unit.

DAE Denoising Autoencoder.

FAO Food and Agriculture Organization.

FMA Foundational Model of Anatomy.

GUI Graphical User Interface.

JAR Java ARchive.

KG Knowledge Graph.

LOD Linked Open Data.

NCI National Cancer Institute Thesaurus.

NLP Natural Language Processing.

OAEI Ontology Alignment Evaluation Initiative.

OWL Web Ontology Language.

PCA Principal Component Analysis.

RAM Random Access Memory.

RDF Resource Description Framework.

RDFS Resource Description Framework Schema.

SCBOW Siamese Continuous Bag of Words.

SG Skip-Gram.

SNOMED CT Systematized Nomenclature of Medicine Clinical Terms.

TF-IDF Term Frequency-Inverse Document Frequency.

UBERON Uber Anatomy Ontology.

UMLS Unified Medical Language System.

URI Uniform Resource Identifier.

Chapter 1

Introduction

1.1 Motivation

In recent decades, biological data experienced considerable complexity, heterogeneity, and volume growth. This development was even more significant in the latest years. A data-driven era emerged due to scientific research advancements, technological development and employment in several science and business sectors, and reduced digital storage costs. Despite the interconnected nature of the data generating domains and similarity of the respective concepts, this produced information usually contains a strict, non-transferable, probably evolving vocabulary, which hinders a possible integrated analytical process. Data mining, which is the process of extracting useful patterns from collections of data resorting to algorithms and techniques from fields such as statistics, machine learning and data warehousing, can take advantage of integrated data to boost knowledge discovery [22]. In order to facilitate analysis and still maintain the detail of the information to posteriorly transform it into useful knowledge, it was needful to structure and connect this growing information.

In 2001, Tim Berners-Lee envisioned an extension of the World Wide Web, which would supply its content with well-defined meaning. The so-called Semantic Web, had the objective to enhance computer and human cooperation, by defining information in an accessible and comprehensible format for both ends, taking advantage of ontologies [2]. In Semantic Web terms, an Ontology is a file which describes concepts inside a determined domain and relations between them in a formal and standardized scheme. Ontologies would solve the ambiguity of natural language where humans used the same term to refer to different things and vice-versa, allow to conduct automated reasoning upon its content, and make it accessible to everyone by defining each term using URIs. This set of rules to publish and connect structured data around the web would be called Linked Data [3].

This practice is being increasingly embraced. The Linked Open Data (LOD) Project[1] is the result of a

[1] https://lod-cloud.net/

considerable effort in building a web of information based on the adoption of the Linked Data principles. Its illustration in Figure 1.1 displays the significance of the *Life Sciences* domain inside this representation of knowledge, which constitutes data related to biological, biochemical, drugs, and species and their habitats concepts [53].

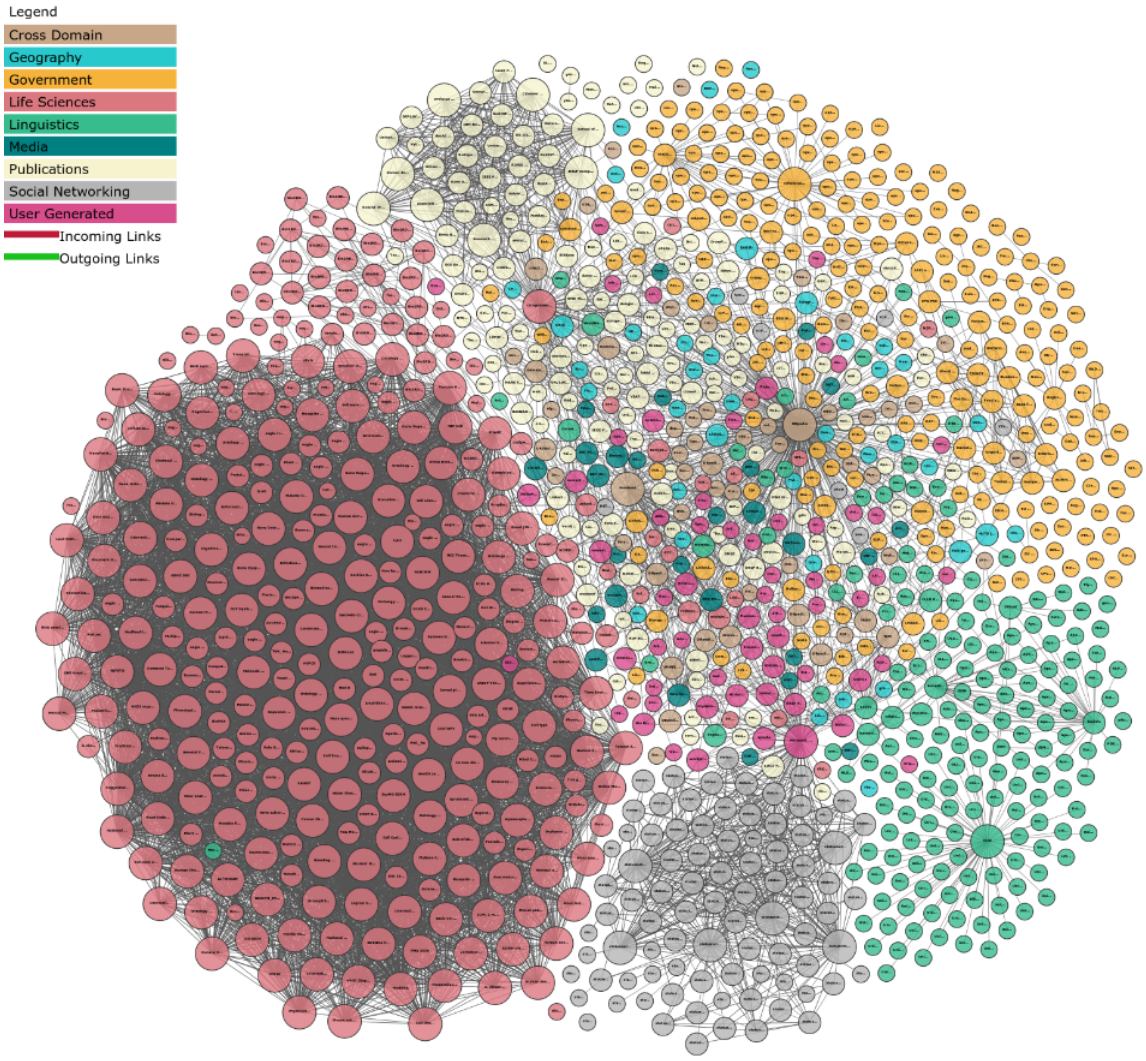

Figure 1.1: The LOD Cloud, 2017.

Real-world entities and relations between them can be described (annotated) by concepts from multiple ontologies in a directed graph structure denominated Knowledge Graph (KG) [16, 27]. Naturally, distinct domains are interconnected and contain intersections. This originates a setback for ontologies because although they represent a standard inside a particular domain, the description of similar concepts in different terms inside distinct ontologies dissolves the intended semantic standard, and the human id-

iosyncrasy obstacle to semantic data structuring emerges once again. The approach to circumvent this issue is the process of discovering correspondences between those concepts and establishing links between them, known as KG Matching.

There are over 800 distinct biomedical KGs in BioPortal [45], many with overlapping domains but practically no established links between them, preventing the integrated analysis of heterogeneously annotated data. Creating alignments between them manually requires domain specialized human resources and is a very demanding and impractical task. Consider as an example, the manual alignment of the Chinese Agricultural Thesaurus (CAT) ontology of around 60,000 concepts with the Food and Agriculture Organization (FAO) of the United Nations Thesaurus, AGROVOC ontology, of around 25,000 concepts took seven person-years of unwearying work [35, 59]. This process is even more unfeasible for large biomedical ontologies like Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) [15], which contains a rich and complex vocabulary, heterogeneous data, and more than 300,000 concepts at the time of writing this dissertation.

At this point, it should be comprehensible that the KG matching task must be automated. Since 2004, a competition named Ontology Alignment Evaluation Initiative (OAEI)[2] is organized yearly to assess the performance of state of the art matching systems on specific matching tasks, including sizeable biomedical KGs. Although most of the systems steadily improved over the last years, their performance on the biomedical tasks has generally stagnated. As we can see in Table 1.1, the best F-measure values over the last four years are nearly identical.

Test Case	2016	2017	2018	2019
Human Anatomy -> Mouse Anatomy	0.943	0.943	0.936	0.943
FMA -> NCI small fragments	0.931	0.930	0.933	0.933
FMA -> SNOMED small fragments	0.825	0.835	0.835	0.835
NCI -> SNOMED small fragments	0.797	0.804	0.801	0.818

Table 1.1: OAEI Biomedical Ontologies test cases best F-measure results per year since 2016.

Developing KG matching strategies for such a complex area like biomedicine is a challenging task for various reasons. While the high quantity of entities poses a computational and visualization difficulty, its sophisticated vocabulary with many labels and synonyms creates a challenge for algorithms like lexical matchers, which need to distinguish and select labels with different degrees of relevance [20]. In addition, because "even science isn't an exact science", the same biomedical domains may be defined with different points of view, causing logical incoherences when trying to match two ontologies (e.g. in FMA ontology, *Fibrilar Actin* becomes a subclass of both *Anatomic Structure System* and *Substance and Gene Product* in NCI ontology, which are disjoint classes [47]). Lastly, biomedical ontologies have evolved in semantic richness, inheriting complex axioms. For instance, "human patient and (has Age some float [>= 8])

[2] http://oaei.ontologymatching.org/

participant in WHO standard treatment for human brucellosis in adults and children eight years of age and older". Usually, ontology matching systems focus mainly on taxonomic relations or do not discriminate types of relations [10].

In the last few years, KG matching systems have developed increasingly complex strategies to handle the challenges in matching biomedical KGs [20]. These strategies include highly complex combinations of algorithms to explore the lexical and structural components of the ontologies, and the use of external knowledge in the form of other biomedical ontologies and thesauri. These efforts have been in large part driven by the OAEI biomedical tasks, where benchmarks are made available (both ontologies and reference alignments). The difficulties in producing a generalizable KG matching approach were made clear with the introduction of blind tasks (for which no reference alignment exists). Although systems were able to find many of the most straightforward mappings, they failed to find mappings belonging to a curated set generated by experts [24].

1.2 Goals

Even though traditional matching techniques are engineered to withstand most entanglements of biomedical KGs, they are stagnating. With that in mind, the main objective of this dissertation was to improve state of the art in Biomedical KG Matching by developing a novel and more generalizable methodology, based on Neural Embeddings techniques to develop new similarity measures for comparing entities between two distinct KGs. Embeddings provide a way to compare entities without the need to produce complex algorithms, but both entities need to be represented in a common semantic space. This is the case when two entities are a part of the same text corpora [42], or part of the same KG [5]. Both unsupervised and supervised strategies can be used to compare entities, the former typically based on vector similarity and the latter based on more sophisticated machine learning approaches.

The hypothesis guiding this work was that neural embeddings could be beneficial on the matching process in two relevant fronts: first, by providing semantic information which may be lacking in ontologies, using appropriate contextual text corpora; second, by interpreting the graph structures and information inside the ontologies, which is not easily comprehended by traditional matching approaches. This work tackles a number of challenges:

1. In the first challenge, we face the obstacle of linking the KG entities to the text corpus entities, and then how to employ word embeddings trained on external corpora to compare ontology classes that may have multiple multi-word labels. Corpus-based embeddings are typically employed to compare words or documents, not entities that can be described by different multi-word terms.

2. In the second challenge, we have two distinct semantic spaces, i.e., each of the KGs to match, so to enable graph embeddings computation, we must first solve the difficulty of creating a unified semantic space.

3. Finally, several state-of-the-art embeddings based KG matching approaches employ supervised learning, which limits their applicability to domains where a reference alignment is not available. Therefore, we tackle the challenge of circumventing the need for a reference alignment in supervised strategies.

The main focus of the work is on the ontology matching component of KG matching, first because this is the first challenge in KG matching, but also because many of the developed approaches can be in principle extensible to instance matching. The methods and approaches were evaluated using the benchmarks provided by the OAEI biomedical test cases.

1.3 Document Structure

The current chapter provides introductory exposure to the research challenges to be tackled with the proposed hypothesis and the objectives of this dissertation. Chapter 2 explores the field's state of the art concepts, techniques and relevant work developed until this date. Chapter 3 describes the overall methodology's steps and architecture. Chapter 4 presents the data utilized to assess the methodology's performance. Chapter 5 describes more in detail the implementation of the constructed approaches and exposes and debates the obtained results. Chapter 6 gives a rundown over the main conclusions of this work and indicates a set of directions for future work.

Chapter 2

State of the Art

This chapter introduces the essential concepts and techniques to comprehend the presented work, and developed state of art procedures in these research fields.

2.1 Knowledge Graphs

Tim Berners-Lee introduced the concept of Semantic Web as "an extension of the current web, in which information is given well-defined meaning, better-enabling computers and people to work in cooperation" [2]. For this idea to be materialized, it required the existence of structured collections of information, and sets of inference rules that they can use to conduct automated reasoning. This web of data needed to be accessible to everyone, standardized in a meaningful way to both humans and computers, and decentralized for the sake of scalability and not being a single point of failure.

For those reasons, the published knowledge would need to comply with a set of best practices to publish and connect structured data around the web, known as Linked Data [3]. These practices would include defining a concept using an Uniform Resource Identifier (URI) and structuring information related to it in a format called Resource Description Framework (RDF). RDF is a standard model for defining relationships between any two pieces of data around the Linked Data. The RDF format was posteriorly extended to Resource Description Framework Schema (RDFS), which included a type system. This enlarged the language with semantic capabilities, allowing the definition of classes, groups which concepts could belong to, subclasses and subproperties, enabling the definition of hierarchies of classes and properties [38].

A RDFS statement, also known as a triple, expresses a relation between two resources in a subject-predicate-object format. As shown in Figure 2.1, those two resources can be two concepts, so both subject and object are both URIs, or a concept and an inherent property, hence represented by a subject URI and a literal string object. The predicate is also an URI and indicates how the subject relates to the object.

Later on, a more robust representation, which addressed a set of RDFS's limitations, was proposed.

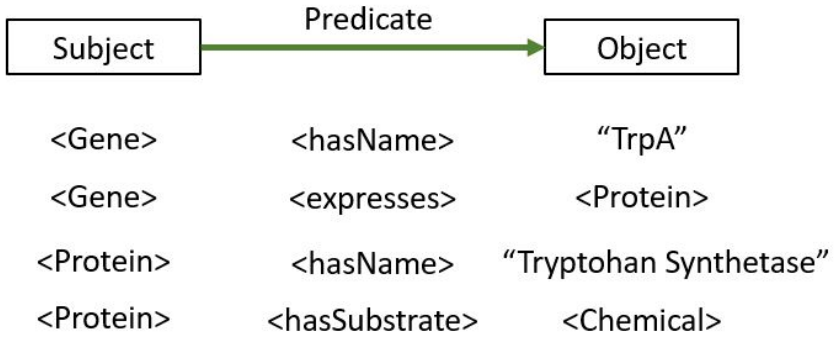

Figure 2.1: Example of RDFS statements. URI prefixes are ommited for the sake of plainness.

Web Ontology Language (OWL) is the current standard language for representing ontologies, which are formal documents that define the schema of knowledge for various domains, describing concepts and relations between them. OWL contains a more substantial expressibility than RDFS and therefore allows to phrase more complex semantic relations like cardinality (e.g. any person has one and only one biological mother), disjointness (e.g. a person is either a smoker or non-smoker) and transitivity (e.g. if B is bigger than A and C is bigger than B, then C is bigger than A) [39]. This added logic and semantic richness created the possibility of having software agents inferring content inside ontologies, making another significant step in the process of making machines "understand" the content inside the Semantic Web [2].

The Semantic Web encouraged a graph-oriented representation to model such complex and interconnected knowledge, KGs. These structures allow to describe real-world entities and their interrelations, define classes and relations of entities in a schema and cover various topical domains [46]. That being so, a KG can be understood as the composition of the standardized structure of the domain, and the individuals/instances annotated by that schema. Figure 2.2 represents an example of a KG, constituted by a portion of the Gene Ontology [11] which annotates an instance, the P69905 hemoglobin.

2.2 Knowledge Graph Matching

KG matching is the process of finding relations between two distinct KGs and can be decomposed in two parts, ontology matching and instance matching. While ontology matching focuses on aligning the schema of the knowledge graphs that are used to describe their domain, instance matching searches for related individuals based on their properties.

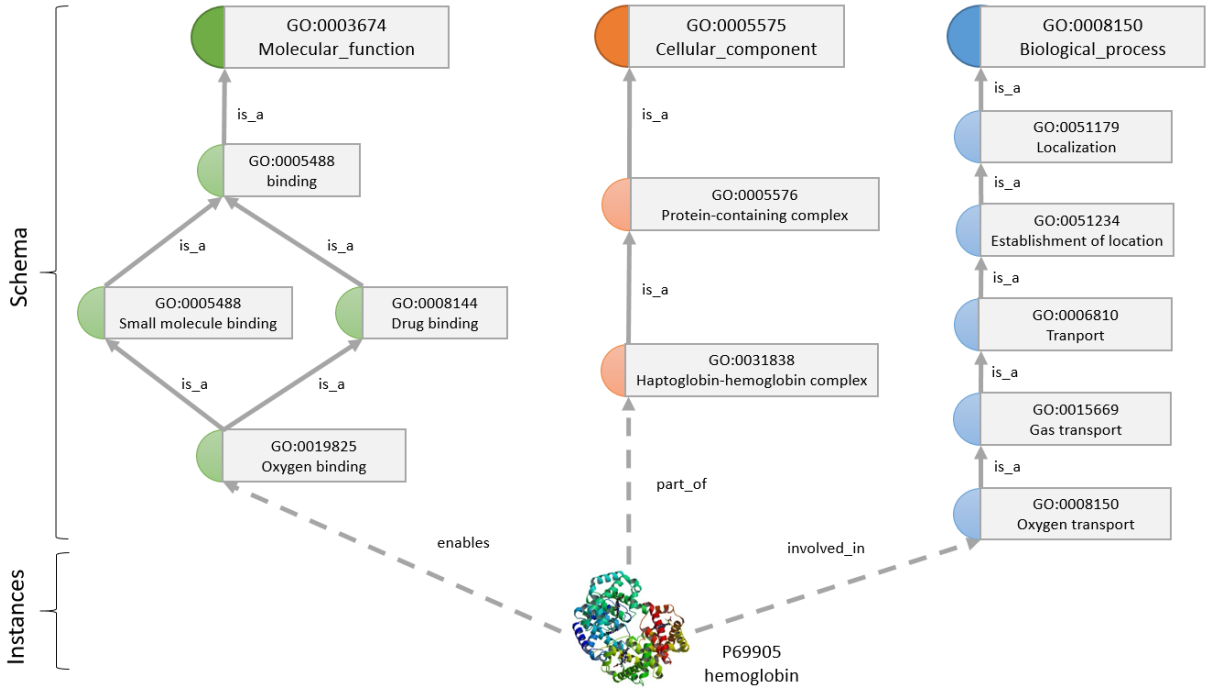

Figure 2.2: Knowledge Graph representation of part of the Gene Ontology.

2.2.1 Ontology Matching

Ideally, ontologies would be built in such a manner that no ontology should contain overlapping content with others. This way, one term was guaranteed to contain only one specification. However, in several cases, domains are represented by multiple ontologies. Thus, the Tower of Babel effect which ontologies were trying to solve arises once again.

Ontology Matching can be defined as "the process of finding relationships or correspondences between entities of different ontologies" [17]. The following ontology matching formalization is based on the work in [55]. Let O_1, O_2 represent a pair of distinct ontologies. Let e_1, e_2 be entities of O_1 and O_2, respectively. Let also r be a relation (e.g. equivalence (=), more general ($\sqsupseteq$), disjoint ($\perp$)) between e_1 and e_2 and c a real number which indicates the degree of confidence that a certain relation r exists, such that $0 \leq c \leq 1$. A matching task determines an $alignment$ A', which is composed of correspondences between O_1 and O_2. An $alignment$ is a set of correspondences (or mappings) between entities from matched ontologies. A correspondence between two ontologies can be described as tuple:

$$< e_1, e_2, c, r >$$

Alignments contain a cardinality, which indicates how many entities are related to each side. Cardinalities can be of: 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) or n:m (many-to-many). In this work, only 1:1 equivalence correspondences will be sought after, since equivalent mappings already

pose a complex task for large ontologies. An alignment example can be seen in Figure 2.3.

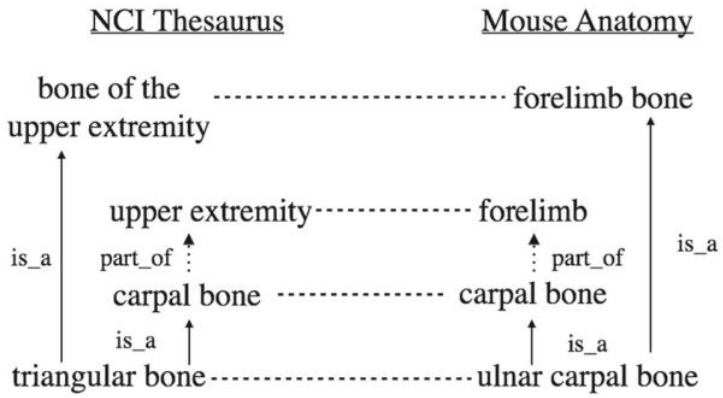

Figure 2.3: Correspondences between the NCI Thesaurus and the Mouse Anatomy Ontology. The dashed horizontal lines correspond to equivalence mappings between both ontologies. Extracted from Kolyvakis et al. [31].

The matching task does not necessarily start from scratch. An input *alignment* A can be used as a starting point to find new correspondences, analysing entities related to the already matched ones, which is called alignment extension. This procedure can reduce computational costs significantly since it does not perform a brute force search for correspondences. That is, attempting to match all pairs of entities. Additionally, external parameters (e.g. weights, thresholds) and external resources (e.g. domain-specific thesauri or knowledge) can provide some guidance in the matching process.

There are multiple techniques which can be used to perform the matching task. These techniques can be separated into two major categories according to the granularity of interpretation of ontologies, element-level and structure-level [17]. Element-level matching techniques analyse entities in isolation, meaning, ignoring the relations to other entities they are connected to. Most common element-level techniques are:

- **String-Based Techniques**: these take advantage of the structure of names and descriptions of ontology entities as sequences of letters and calculate a distance between those sequences, applying string similarity functions. The more similar the strings are, the more likely they describe the same entity.
- **Language-Based Techniques**: they interpret entity names as words in a particular natural language (e.g. English, Latin), enabling the usage of NLP strategies to extract meaningful terms and relations from text. Language-based methods can be divided into algorithms which only use the input texts from the ontologies to match, and strategies which employ external resources such as lexicons, thesauri and corpora.

- **Constraint-Based Techniques**: methods that study the internal structure and respective constraints of ontology entities, that is, the details of their properties such as data type and cardinality of attributes. This information can be used aside or complementing their names and identifiers.
- **Informal Resource-Based Techniques**: when entities from two distinct ontologies annotate the same informal resources such as pictures, these connections can be used to deduce relations between those ontologies using data analysis and statistical procedures.
- **Formal Resource-Based Techniques**: external ontologies can also be exploited to assist in the matching process by establishing bridges between the ontologies to match. Domain-specific ontologies are the most used in this type of technique since they are the most effective in establishing common ground between matching ontologies.

Structure-level techniques discover correspondences according to the analysis of how entities relate to each other in a structure. Some examples of structure-level techniques are:

- **Graph-Based Techniques**: algorithms which perceive the ontologies to match as labelled graph structures. The premise of these techniques is that the similarity between a pair of nodes from two ontologies is associated with the positions and neighbourhoods of those entities.
- **Taxonomy-Based Techniques**: similar to graph-based techniques, except they only consider specialisation relations. The intuition behind this variant is that specialization relations like *part_of* or *is_a* connect similar terms. Therefore, neighbours of equivalent terms which are connected through specialisation relations might also be similar as previously shown in Figure 2.3.
- **Logic-Based Techniques**: logic-based or semantically grounded techniques are methods based on reasoning. These methods find new correspondences by inference and evaluating propositional satisfiability. Being a deductive type of technique performing an inductive task such as ontology matching, they are only reasonably successful when extending an initial alignment.
- **Instance-Based Techniques**: these methods compare sets of instances of ontology classes to determine if the concerned classes are a match or not. They employ similar data analysis and statistical procedures to informal resource-based techniques.

2.2.2 Instance Matching

Similarly to ontology matching, instance matching is the process of discovering correspondences between two real-world entities of two different knowledge sources through the calculation of a similarity value between them [8]. Instances (or individuals) can appear heterogeneously represented in different documents and annotated with terms from multiple ontologies, depending on the necessary concepts to

describe them. Therefore, establishing links between ontologies on the schema level is crucial for instance matching, since those relations will refine the final similarity value between two instances.

2.3 Neural Embeddings

An embedding, in essence, is a mapping of a discrete categorical variable in a vector of real values. Neural network embeddings or neural embeddings are learned distributed vector models of categorical variables in a low dimensional vector space. In other terms, it is a predictive learning-based model which takes advantage of a neural network to embed variables in n-dimensional vectors. Although the learning step is based on prediction, the model can be interpreted as unsupervised in the sense that humans do not need to create labels to feed the learning process. The main advantages of this kind of technique are reducing the dimensionality of highly complex data, being a compact format and enclosing similarity between related data entries [9].

Calculating a value between two terms that represents how similar they are is a core process for the matching task and many NLP applications. Along the years, many similarity metrics were proposed, such as Cosine, Dice, Euclidean, and Jaccard. In vector space models, Cosine Similarity is one of the most popular methods. It measures the angle between two vectors, which can be efficiently calculated as a dot-product of two normalized vectors [34]. Formally, given two vectors $\vec{v}$ and $\vec{w}$ of dimension N the cosine similarity between them can be expressed as:

$$cosine(\vec{v}, \vec{w}) = \frac{\vec{v} \bullet \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{n} v_i \times w_i}{\sqrt{\sum_{i=1}^{n} v_i^2}\sqrt{\sum_{i=1}^{n} w_i^2}} \tag{2.1}$$

Cosine Similarity is also often referenced as Cosine Distance, which is simply $1-$ Cosine Similarity.

2.3.1 Word Embeddings

Mikolov et al. presented [40] two bi-layered neural network model architectures for learning distributed representations, the Continuous Bag of Words (CBOW) and the Skip-Gram (SG), which can be shown in Figure 2.4. The difference between these two models is that given a word in the corpus, a window of its predecessors and successors, while CBOW tries to predict the current word based on the previous and the following words, SG uses the current word to predict the others.

These models are effective to explore relations between single words. For instance, a certain word corpus containing countries names and capitals can be mapped to vectors of continuous numbers using neural embeddings. This type of technique is refered as word2vec. A peculiar condition of the resultant vector space, is that vectors can be added and subtracted effortlessly. With that in mind, being *v[w]* the vector of a word *w* we can observe that the closest vector obtained by making the calculation *v[Paris]* -

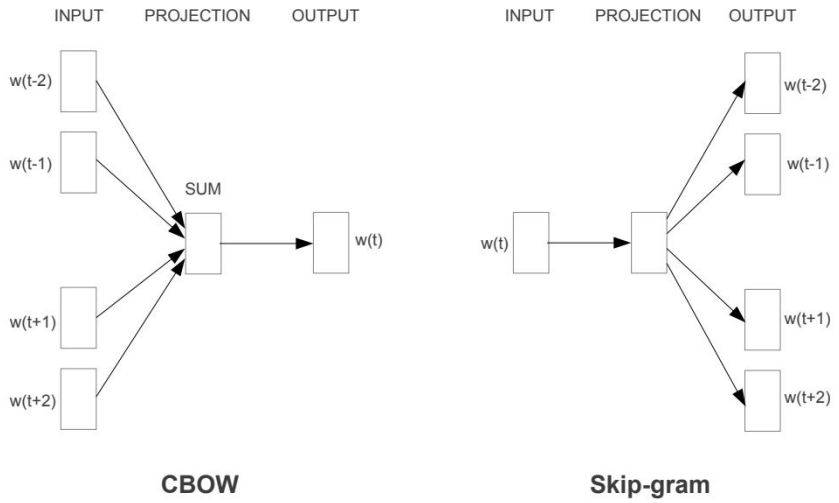

Figure 2.4: The two neural network proposed model architectures, CBOW and SG. Extracted from Mikolov et al. [40]

v[France] + *v[Italy]* is *v[Rome]*. Figure 2.5 shows the model capacity to group terms and learn connections between them without additional supervised learning.

Because of this abilities, this type of model has been sough-after for several Natural Language Processing (NLP) tasks, machine translation and automatic speech recognition [36, 41, 54, 56].

However, many terms in the biomedical domain are expressed by multiple words, such as *Ulnar Carpal Bone* or *Bone of the Upper Extremity*. Two main alternatives are available to enable comparison between these terms. The first one is representing a term by the average of its word vectors.

$$v[Ulnar\ Carpal\ Bone] = \frac{v[Ulnar] + v[Carpal] + v[Bone]}{3} \tag{2.2}$$

That way, the term is represented by a single vector which can be compared to another term. An improved version of this strategy would be multiplying weights to the words vectors according to Term Frequency-Inverse Document Frequency (TF-IDF), a statistical measure of the frequency of a term's use in a corpus [57]. Although this is a popular workaround, word embeddings are not optimized to represent sentences. For instance, the word order in the terms is lost when averaging its vectors. The other possibility, a later proposed more robust model, which could represent segments of text, from sentences to documents named Paragraph Vector, also denoted as doc2vec [33].

2.3.2 Graph Embeddings

Neural embeddings can also be used to represent graph structures. There are several graph embedding approaches which can be organized in groups according to the type of used techniques [6]. These em-

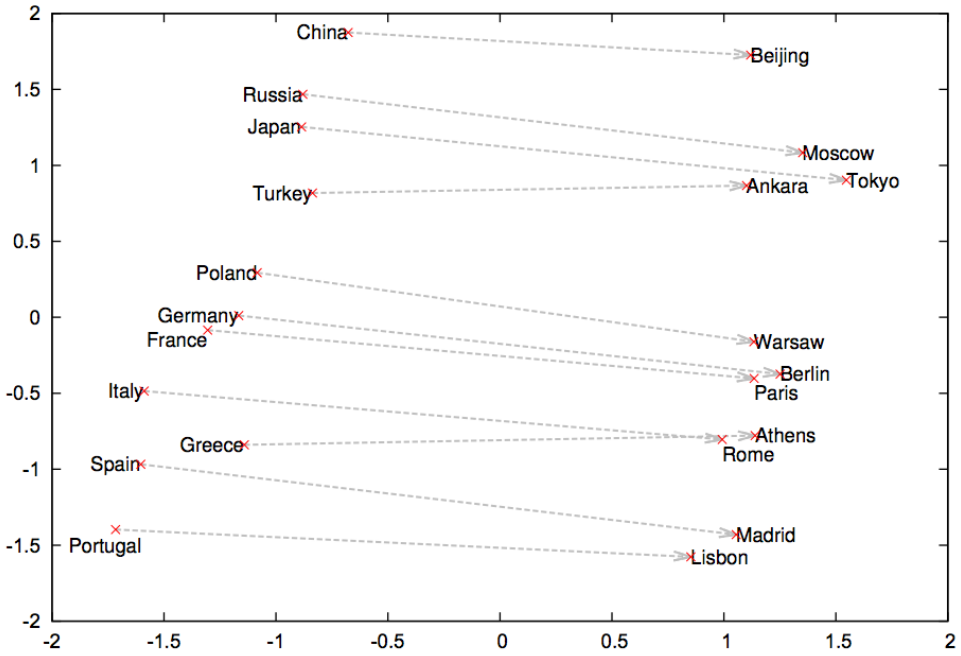

Figure 2.5: Example of Country and Capital Neural Embeddings trained model. Presented in a two-dimensional PCA projection of a 1000-dimensional vector space. Extracted from Mikolov et al. [42].

bedding categories are:

- **Matrix Factorization**: approaches which represent graph properties like node pairwise similarity in a matrix and factorize it to extract the nodes embeddings.
- **Deep Learning**: deep learning based embeddings apply deep neural network models to embed the graph information. This type of technique contains models who receive random walks as input, that is, sequences of visited nodes in a graph walk, or the whole graph structure.
- **Edge reconstruction**: methods that optimize objective functions to predict the edge embedding based on the embeddings of the two nodes linked by the prior. Translating embeddings (TransE), [5] one of the most popular graph embedding strategies, is based on edge reconstruction.
- **Graph Kernel**: approaches that represent whole graphs as vectors where each vector dimension contains information of extracted graph substructures like subgraphs, subtrees or random walks. Two graphs can be compared by computing the inner product of the two correspondent vectors.
- **Generative model**: procedures which embed similar graph nodes next to each other based on their distance in the graph structure or their semantic properties.

Some of the techniques described above are employed in the link prediction problem, which corresponds to estimating the likelihood of a certain unobserved link between two nodes, based on observed

attributes and links of the respective nodes inside a certain graph [23]. In a way, the KG matching task can be expressed as a cross-KG link prediction challenge where we are predicting the equivalence between two nodes of distinct graphs.

Ristoski et al. proposed an approach to represent RDF graphs in an embedding space, RDF2Vec [50]. This procedure generates paths in a graph, using methods like Random Walks and Weisfeiler-Lehman Subtree RDF Graph Kernels [14], to create sequences of entities which are used as sentences to train a language model similar to word2vec, like CBOW or SG. After the train is finished, every entity is represented as a vector of latent numerical features.

2.4 Related Work

In this section, we survey the literature on related work to ontology matching, focusing on the search of neural embeddings applications which could assist in tackling the challenges mentioned in section 1.2. The examined work includes systems evaluated in OAEI biomedical KGs and state-of-the-art KG neural embeddings variants.

Kolyvakis et al. [31] proposed an ontology matching framework named Deep Align[1], based on an extension of the CBOW model named Siamese Continuous Bag of Words (SCBOW) [30]. This extension uses supervised learning to predict sentences occurring next to each other and optimizes pre-trained word embeddings to be averaged and construct high-quality sentence embeddings. This framework also includes a Denoising Autoencoder (DAE) as an outlier detection system to discover misalignments [61]. The reason for this is because word embeddings models tend to learn embeddings that retain both semantic similarity (horse, stallion) and relatedness (horse, harness). Since in the alignment task we are interested in equality matches, only semantic similarity is desired. The DAE learns the inherent properties of the distribution of semantically similar terms to exclude related ones.

This supervised approach surpassed many of the OAEI competing systems, but a direct comparison cannot be made since OAEI competitors are not allowed to employ reference alignments to train models. Moreover, using supervised learning in the biomedical KGs is an issue because there is a substantial lack of training data, in this case, correct alignments. The very purpose of automating the matching task is because it is unrealistic to align KGs manually. Therefore, this kind of system is unscalable in real-word knowledge discovery. Besides, it only takes into account the primary names of entities, which can be throwing away potential synonyms and labels, valuable in the matching process. Additionally, this type of approach struggles with larger ontologies since it represents a quadratic complexity matching computational problem, added to layers of machine learning training time for each task.

[1] https://github.com/prokolyvakis/deep-align

ALOD2Vec Matcher is an ontology matching tool built by Portisch et al. [48] using a RDF2Vec model trained over a web-scale RDF dataset, rich in hypernymy relations, the WebIsALOD [26]. For the alignment process, first and foremost, the tool filters entities with equivalent textual labels. After that, it establishes a link between the remaining entities of the ontologies to match and the entities in the vector space using the textual label and calculates the similarity between pairs of entities using cosine similarity. This model participated in the OAEI 2018 competition producing average results on small biomedical matching tasks in terms of F-measure, discovering mostly mappings already found using string equivalence between entity labels. Nevertheless, the most concerning aspect of this matcher is the run time. The system is amongst the top three slowest matchers in the respective tasks, which becomes apparent given the dimension of the dataset used to train the model.

Jimenez et al. [29] constructed a mechanism to split the ontology matching task into smaller matching tasks for any matching system, cutting out some of the computational complexity and required memory. The strategy consists of creating a hash dictionary where the key is a set of words, and the value is a set of entity identifiers. Entities of both ontologies to match with words in common are indexed on the same entry. Posteriorly, the entries are divided into k clusters, creating k smaller searching spaces to match. One of the splitting techniques takes advantage of neural embeddings to make more intelligent splits, by computing the average vector of all words inside an entry of the dictionary and using K-Means clustering algorithm to group entities according to the features of the vectors.

While this procedure reduces the complexity of the matching task, it is not helpful in performance on benchmarks. In fact, it reduces the performance slightly due to the reduction of coverage in the matching space, that is, some of the relevant ontology mappings in the original matching task are lost when dividing it into subtasks.

Cai et al. [7] developed a model to perform link prediction across distinct cross-lingual KGs having no connections between them, by training an embedding model using a refined version of translation models such as TransE joint with Jaro-Winkler string metric [62]. A similarity score between entities is calculated using a linear combination of cosine and string similarities. The proposed model attempts to surpass the limitations of translation representation models which attain a low performance on sparse KGs, and of independent representation learning of different KGs that result in no correlation between embedding models, making it impossible to use them in cross-KG tasks such as KG matching.

In this work, the larger and denser KG is given the name of source KG and the smaller and sparser one is named target KG. With that in mind, the target entity labels are translated using Google translate API to the source KG language and if they are semantically equivalent, their embeddings are trained based upon the entities of both KGs. As a result, embeddings from two distinct KGs are learned in a unified vector space, and semantically similar entities are represented close to each other, which increased link prediction performance.

Examining the presented related work, we identify two significant limitations which hinder the progress of KG matching using embedding-based approaches. The first one is the necessity for reference alignments as training data. As previously mentioned, there is a severe lack of reference alignments in biomedical KG matching which makes it unfeasible to use supervised learning approaches on the task. The second limitation is the inability of graph embedding based matching to improve results compared to a string equivalence algorithm, especially when using a large embedding model. Any matching approach should at least improve on string equivalence results, considering it is the simplest approach and often used as a baseline in some benchmarks.

Chapter 3

Methodology

This chapter will present the overall ontology matching methodology based on Neural Embeddings techniques. Given that current KG matching algorithms are highly tailored for present challenges and reached a plateau, we intended to develop more generic approaches, focusing on the hypothesis that neural embeddings could provide auxiliary insight to interpret information better inside KGs. To integrate our embedding methods straightforwardly into the matching process, we benefited from an existing ontology matching system, AML.

3.1 AgreementMakerLight

AgreementMakerLight (AML)[1] is an ontology matching system written in Java, derived from one of the first leading ontology matching systems, AgreementMaker[12]. Most first-generation ontology matching systems were not built with scalability in mind, since the initial matching tasks were relatively small. Therefore, the main goal of building AML was to create a scalable, automated system to tackle the large biomedical ontologies, while maintaining the flexibility and extensibility of the original AgreementMaker [18].

The system is composed of three principal modules, each one responsible for a particular task in the ontology matching pipeline presented in Figure 3.1. The first one is the ontology loading module. This component loads the ontologies stored in OWL files into memory objects which contain hash-based data structures, the *Lexicon* and *RelationshipMap*, to occupy less memory space and be efficiently fetched when needed. While the *Lexicon* keeps the names, labels and synonyms of each term inside an ontology, the *RelationshipMap* stores the described relations between the terms kept in the *Lexicon*. This module is used to load the two ontologies to match, called source ontology and target ontology. Optionally, an external ontology can also be loaded as background knowledge to establish bridges between the input ontologies, or an Alignment in RDF format which may have the role of reference alignment to evaluate

[1] https://github.com/AgreementMakerLight/AML-Project

produced alignments or may be a previously produced alignment to be extended or evaluated by itself.

The second and most important unit is the matching module. This element covers all the matching algorithms and strategies stored in classes called *Matchers*. A matcher produces an alignment object, a collection of mappings (correspondences) between entities of the source and target ontologies with a similarity value above a certain threshold. Since the matching problem has a $O(n^2)$ complexity, using elaborate matching algorithms in all pairs of entities would be very inefficient. Thereby, matching algorithms are divided into two major types of matchers, primary and secondary matchers.

Primary matchers are lightweight algorithms, mostly based on HashMap cross-searches, which achieve a $O(n)$ complexity exploring all matching possibilities. An example of a primary matcher is the Lexical Matcher, which finds entities with the same name. Secondary matchers make non-literal comparisons between terms, thus requiring to compare each entity of the source ontology with all of the entities in the target ontology, taking $O(n^2)$ time. This type of matcher cannot be used to match large ontologies from the ground. With that in mind, secondary matchers extend a previous alignment by exploring the neighbourhood of already mapped entities, creating all the possible pairs of entities from the source and target ontologies which do not conflict with the base alignment, and applying the matching algorithm to calculate a similarity between those pairs. This process of extension is repeated a certain number of iterations or until no more new mappings are found. An example of a secondary matcher is the String Matcher, that creates matches based on a string similarity between the *Lexicon* entries of both entities. AML also has another type of matcher called Rematcher which simply recomputes similarities of a given alignment according to its own algorithm. The system has the ability to choose different matchers depending on the properties of the ontologies to match. Furthermore, AML has multicore implementations for every matcher that could profit from running mapping tasks in parallel, taking advantage of modern CPUs with several cores [20].

The third module of the matching pipeline is alignment filtering. As the name suggests, this component removes some of the troubling mappings from an alignment. Problematic mappings can appear for two primary motives. One of them is cardinality conflicts, where a class of one ontology is mapped with multiple classes of the other ontology where it should only be mapped with one. The other is logical conflicts, where at least two mappings cause the matching ontologies to become logically unsatisfiable when merged using those mappings. AML contains a Selector, which is an algorithm which excludes competing mappings from an alignment to obtain the desired one-to-one cardinality, preserving the ones with higher similarity. There are three types of selection. Strict selection allows no conflicts in the alignment. Permissive selection only allows conflicts of mappings with equal similarity. Hybrid selection allows a cardinality of two for mappings with similarity above 0.75. The system also contains a Repairer algorithm accountable for determining logical inconsistencies for classes or properties, finding the mappings causing those inconsistencies, and removing or modifying those mappings according to a set of repairing rules [52]. These tools contribute to the production of a high-quality final alignment.

The AML framework can be used in different ways. The most flexible way it to program a runnable Java class inside the Eclipse project, using AML's API, to benefit from available tools and features.

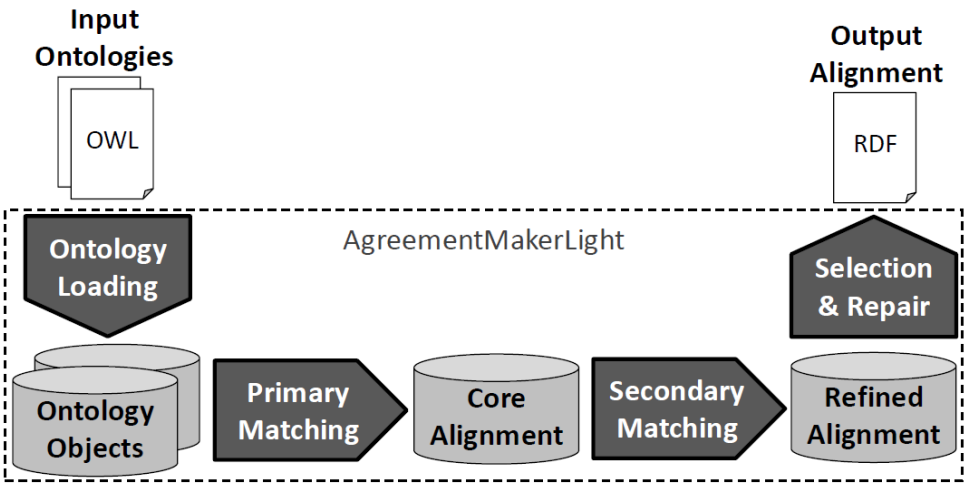

Figure 3.1: AML ontology matching pipeline. Extracted from Faria et al. [19].

Another method is by executing its JAR file on a command line with arguments and a configuration file. There is also a more intuitive and visual alternative for inexperienced users which is a GUI. Beyond the usual loading, matching and filtering capabilities, the provided GUI allows to edit an alignment by adding or removing mappings as desired, and also visualize the graphs of the mappings and its neighbours on both ontologies [37].

AML has obtained the highest results in OAEI 2019 anatomy, conference (four out of five test cases), interactive matching, large biomedical ontologies (five out of six test cases) and biodiversity and ecology tracks [21].

3.2 Overall Methodology

Contrasting with classical methodologies, the development of procedures in this work was more exploratory and iterative. The performance of each strategy was evaluated after its design to determine the following course of action.

Since the key goal of this dissertation was to create a generalizable method and avoid overengineering, our primary approach was to test plain cosine distances to compute a similarity value between entities of the two KGs to align. The nature of used methods includes word embeddings, graph embeddings and combinations between both embedding types and string similarities. On the other hand, it was also desired to compare the results of these general strategies with a more robust model, optimized for these tasks, to study the performance divergence between approaches, but without resorting to unscalable supervised learning methods. For this matter, we proposed an unsupervised version of the Deep Align framework [31].

Despite cosine similarity being efficient to compute, the large computational complexity of biomed-

ical KG matching still prevents us from attempting all-versus-all comparisons. The AML system has shown to be an easy to operate tool with suitable modularity. Therefore, our proposal integrates an embedding matching strategy step inside AML's regular matching pipeline, interacting with required external resources like scripts and embeddings models. This step follows up an initial AML matching strategy composed of any of its algorithms and is continued by a selective filter to produce a final alignment, as shown in Figure 3.2.

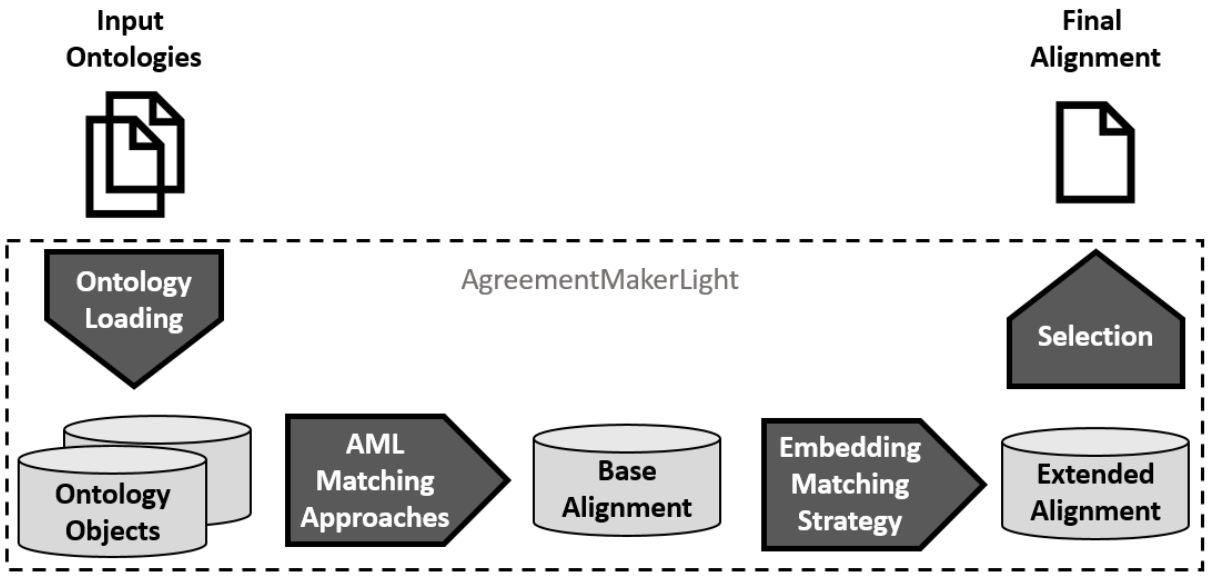

Figure 3.2: Embedding Matching Strategy integrated in the AML Pipeline.

3.3 Evaluation

The performance of a strategy is evaluated by comparing the produced alignment with a ground truth, in this case, the reference alignments. The metrics used to assess the performance of the procedures were precision, recall and F-measure, which is a harmonic mean of precision and recall. These metrics can be expressed by

$$\text{Precision} = \frac{\text{Number of correct mappings}}{\text{Total number of mappings in produced alignment}} \tag{3.1}$$

$$\text{Recall} = \frac{\text{Number of correct mappings}}{\text{Total number of mappings in reference alignment}} \tag{3.2}$$

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

Although the main objective is generally to improve F-measure, in this work we're particularly interested in increasing recall to discover new mappings which weren't captured with other matching strategies.

3.4 Comparison with State of the Art Strategies

Since AML is the state of the art system with overall best results on OAEI large biomedical ontologies track and contains multiple matching algorithms that can be used and combined as necessary, we used some of its procedures to compare our work with. Three AML matching strategies with varying degrees of sophistication were chosen to compare the performance of developed methods. The first one, Lexical Matcher algorithm is the simplest. The second one is the Automatic Matching algorithm. The third is Full AML which constitutes all the available AML resources combined.

3.4.1 Lexical Matcher

The Lexical Matcher algorithm finds entities with the same literal name. It is one of the most efficient and straightforward algorithms which only requires iterating over the names of one of the ontologies once. For two ontologies, *source* and *target*, the algorithm can be described as follows:

Algorithm 1: Lexical Matcher Algorithm

```
set thresh = matcher threshold;
set A = empty alignment;
set list = names(source);
foreach name in list do
    if target contains name then
        set sourceList = sourceClasses(name);
        set targetList = targetClasses(name);
        foreach sourceClass in sourceList do
            set weightSource = weight(name, sourceClass);
            foreach targetClass in targetList do
                set similarity = weightSource * weight(name, targetClass);
                if similarity >= thresh then
                    add (sourceClass, targetClass, similarity) to A;
                end
            end
        end
    end
end
```

The weight of each name in the final similarity depends on the provenance of that name, meaning, whether the name represents the local name, a label or a type of synonym. This algorithm serves as an effective baseline since it finds the most basic mappings between ontologies, thus, any valuable matching strategy should at least surpass this algorithm.

3.4.2 Automatic Matching

The Automatic Matching is the algorithm which combines AMLs capabilities into one proficient matching algorithm. Given that the system contains over twenty matching algorithms and various filtering tools such as selectors and repairers, the Automatic Matching works as a fine-tuned control unit which chooses which matching and filtering algorithms are relevant to use, according to the nature of the ontologies to match. Some of the relevant properties in the decision making are the number of languages, the number of entities and graph connectivity in each ontology.

For instance, when matching OAEI biomedical ontologies of intermediate size, the Automatic Matching algorithm opts for the following pipeline: first, it runs Lexical Matcher and Word Matcher and adds discovered mappings to an empty alignment; then, it extends the alignment with the string matcher; after that, runs Spaceless Lexical Matcher and Thesaurus Matcher and adds non-conflicting mappings to

the previous alignment; finally, it uses a Cardinality Selector with a Hybrid Selection Type to filter the alignment and runs the Repairer algorithm.

3.4.3 Full AML

The full AML matching algorithm corresponds to a version of Automatic Matching which contains all the features in the variant described above but also includes the ability to use external ontologies as background knowledge. AML takes advantage of this external knowledge by using a Mediating Matcher algorithm which establishes connections between the source and the external ontologies, and between the target and the external ontologies, using a lexical matcher approach, meaning, by creating mappings between entities with names in common. Then, it iterates over both mapping collections, and if a source entity and a target entity are connected to the same external mediator entity, they are acknowledged as a mapping.

This procedure can be very beneficial when the external ontology contains several synonyms for each entity which are not contained in the intersection of the source and target ontologies. However, it creates a dependency on the existence of valuable external ontologies. Not only these ontologies require numerous synonyms, but they also need to address the domains that are being matched in detail. Since a large portion of the benchmark biomedical ontologies addresses anatomy, AML employs the Uber Anatomy Ontology (UBERON) [44], rich in synonyms, as background knowledge for these matching tasks.

The motive to distinguish both variants is partially to provide a fairer comparative strategy which does not resort to selective external knowledge, but also compare the performance gain of this external knowledge with our procedures which also provide outside knowledge to the matching process.

The Mediating Matcher can be more precisely described as follows:

Algorithm 2: Mediating Matcher Algorithm

```
set thresh = matcher threshold;
set maps = empty alignment;
set sourceNames = names(source);
set targetNames = names(target);
set sourceMapsDict = lexicalMatchWithBackground(sourceNames);
set targetMapsDict = lexicalMatchWithBackground(targetNames);
set reverseTargetMapsDict = reverseDict(targetMapsDict);
foreach sourceClass in keys(sourceMapsDict) do
    set mediatorValues = sourceMapsDict.get(sourceClass) foreach mediatorClass in
     mediatorValues) do
        if reverseTargetMapsDict contains mediatorClass then
            set targetValues = reverseTargetMapsDict.get(mediatorClass);
            foreach targetClass in targetValues do
                set similarity = min(sim(sourceClass, mediatorClass), sim(mediatorClass,
                 targetClass));
                if similarity >= thresh then
                    add (sourceClass, targetClass, similarity) to maps;
                end
            end
        end
    end
end
```

Chapter 4

Data

As previously mentioned, it is undoable to align biomedical KGs manually. Therefore, the amount of alignments to evaluate KG matching systems is scarce. In this chapter, we present a data source of benchmarks for ontology matching and the datasets used in this dissertation.

Ontology Alignment Evaluation Initiative (OAEI) is a coordinated international initiative that assesses the performance of ontology matching systems on yearly evaluation events since 2004. OAEI consists of various tracks representing groups of ontologies of particular domains and reference alignments between them. The participating systems are evaluated by comparing the mappings in their produced alignments with the ones on the reference alignments.

In this work, the OAEI 2019 edition test sets will be used to assess the developed methodology's performance. The relevant tracks for our evaluation are "anatomy" and "Large Biomedical Ontologies (largebio)". "Anatomy" track is composed by the Adult Mouse Anatomy [25] and a portion of the NCI Thesaurus [13], respective to the description of the human anatomy. This small test case is the only one manually aligned, hence the only gold standard for evaluating matching systems.

The "largebio" track reference alignments were produced based on the extraction of relations from the Unified Medical Language System (UMLS) Metathesaurus [4]. UMLS, a set of files and software, is the most extensive attempt to integrate independently developed ontologies and thesauri, such as Foundational Model of Anatomy (FMA) [51], National Cancer Institute Thesaurus (NCI) [13], and SNOMED CT [15] which were the chosen ontologies for this track. NCI and SNOMED CT are more broad ontologies but still describe anatomy like FMA, which is one of the fields that bonds them together. Even though the connections between ontologies are accurate in lexical terms, the semantic aspects are often disregarded, resulting in some of the mappings violating the documents' reunion logical integrity [28]. For this reason, mappings in the reference alignments which introduce logical inconsistencies are flagged and ignored during the evaluation phase to avoid penalizing systems that either do or not a repair process to remove inconsistent mappings. Additionally, in this track, some of the ontologies contain large amounts of classes and take a great computational effort to match. Some of the datasets consist of smaller fragments of pre-existent ontologies with the most relevant matching knowledge, created to reduce the

search space of those challenges.

Furthermore, these ontologies entirely focused on the conceptual schema and are devoid of instances. Auxiliary aliases were given to the ontologies to address them more easily. A description of the ontologies and reference alignments is present on Tables 4.1 and 4.2 respectively.

Ontology	Alias	OAEI Track	Classes	Properties
Adult Mouse Anatomy	mouse anatomy	anatomy	2743	3
Human Anatomy (NCI fragment)	human anatomy	anatomy	3304	2
FMA small overlapping nci	FMA small nci	largebio	3696	24
FMA small overlapping snomed	FMA small snomed	largebio	10157	24
FMA whole ontology	FMA whole	largebio	78988	54
NCI small overlapping fma	NCI small fma	largebio	6488	63
NCI small overlapping snomed	NCI small snomed	largebio	23958	82
NCI whole ontology	NCI whole	largebio	66724	190
SNOMED extended overlapping fma nci	SNOMED extended	largebio	122464	55
SNOMED small overlapping fma	SNOMED small fma	largebio	13412	18
SNOMED small overlapping nci	SNOMED small nci	largebio	51128	51

Table 4.1: OAEI test dataset ontologies.

Reference Alignment	Source Ontology (by alias)	Target Ontology (by alias)	Number of Mappings (logically consistent)
Human-Mouse Anatomy	human anatomy	mouse anatomy	1516
FMA2NCI UMLS	FMA whole or FMA small nci	NCI whole or NCI small fma	2686
FMA2SNOMED UMLS	FMA whole or FMA small snomed	SNOMED extended or SNOMED small fma	6026
SNOMED2NCI UMLS	SNOMED extended or SNOMED small nci	NCI whole or NCI small snomed	17210

Table 4.2: OAEI reference alignments.

Chapter 5

Developing Embeddings based Matching Strategies

This chapter elaborates on the exploratory and iterative development process of embedding strategies for ontology matching. The performance of each strategy was evaluated on three of the small matching tasks of the Large Biomedical Ontologies OAEI track and posteriorly compared to the state of the art approaches, provided by AML and are presented in Table 5.1. Due to the fact that we used the same three small fragments for all tests, the names of the test cases were simplified for the sake of readability. Instead of "mouse anatomy -> human anatomy", we have "Anatomy". Instead of "FMA small nci -> NCI small fma", we have "FMA -> NCI". And for "FMA small snomed -> SNOMED small fma" we have "FMA -> SNOMED".

	Anatomy			FMA -> NCI			FMA -> SNOMED		
Comparative Strategy	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	81.8%	88.7%	97.9%	62.2%	76.0%
Automatic Matching	96.1%	83.6%	89.4%	96.8%	86.8%	91.5%	92.8%	74.0%	82.3%
Full AML	95.0%	93.6%	94.3%	95.8%	91.0%	93.3%	92.3%	76.2%	83.5%

Table 5.1: The three AML provided comparative strategies, evaluated on three small matching tasks of the Large Biomedical Ontologies OAEI track.

Each evaluated strategy is tested with every filter Selection Type (Strict, Permissive and Hybrid) but only results with Strict Selection will be present in this document. A full version of the results is available in the following google drive link.

5.1 Embeddings Similarity

The most intuitive procedure of incorporating plain embedding similarities in AMLs matching pipeline is by developing a secondary matcher for the system. This way, we can take advantage of its extension mechanism and integrate the similarity values directly as the neighbourhoods of mappings are being traversed. The extension is then filtered by a selector to resolve cardinality concerns.

5.1.1 Word Embedding Matcher

Training a neural embeddings model is expensive in terms of time and computational power. Therefore, some pre-trained word models were used within this work. Not all models are suitable for this challenge, for the reason that biomedical knowledge has a very particular vocabulary. Expressions like *macrothrombozytopenia* or *paracoccidioidomycosis* should be included in the models in order to allow comparisons of the respective terms. Moen et al. [43] published sundry NLP resources[1], including four word2vec models based on large biomedical scientific literature text corpora from PubMed[2] and PubMed Central[3] and an English Wikipedia dump. These models apply a SG model with a window size of 5, trained with hierarchical softmax and a frequent word subsampling threshold of 0.001 to create the 200-dimensional vector models, represented in Table 5.2.

Pre-Trained Model Name	File Size (GB)
PubMed	1.8
PMC	1.9
PMC + PubMed	3.2
PMC + PubMed + Wikipedia	4.3

Table 5.2: Word2vec pre-trained models.

A Word Embedding Secondary Matcher was built on two components, one in Java and the other in Python. The Java constituent is responsible for extending the received alignment by exploring the graph connections of the two ontologies, creating a list of candidate mappings to calculate similarity and writing them to a text file, calling the python component to compute the embeddings similarity between the required term pairs, reading the similarities from another text file, calculating the confidence of the mappings based on the maximum similarity between terms and keeping the mappings above a certain threshold.

The python element uses an open-source library named Gensim [49] which allows to create and manipulate embedding models such as word2vec. It is responsible for loading a pre-trained word2vec model,

[1] http://bio.nlplab.org/
[2] https://pubmed.ncbi.nlm.nih.gov/
[3] https://www.ncbi.nlm.nih.gov/pmc/

reading the pair list of terms from the text file, averaging the words vectors in each term, calculating the cosine similarity between each pair of terms, and writing the similarities in another text file. By default, if a word is not contained in the embeddings model vocabulary, the cosine similarity between terms is considered zero. An overall representation of the pipeline is shown in Figure 5.1.

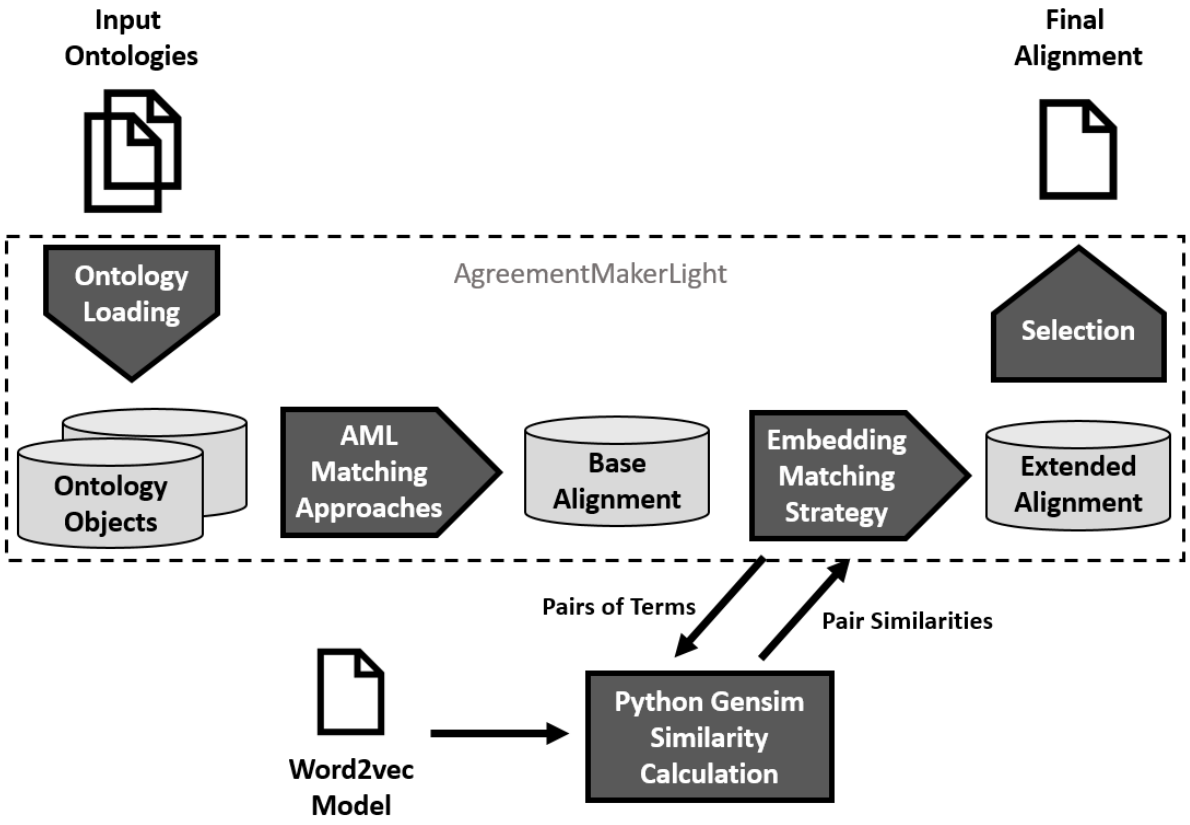

Figure 5.1: Word Embedding Matcher Pipeline.

An initial test was carried out to compare the performance of the four pre-trained word models. The test consisted of extending the Lexical Matcher with the Word Embedding Matcher and filtering the extension by applying a Selector. The threshold of the Word Embeddings Matcher and Selector was equal and obtained by experimenting with different values and choosing one that benefited recall, with the common sense of not sacrificing precision excessively. The produced results of the various models with a Strict Selector, along with the Lexical Matcher baseline are presented in Table 5.3.

In two out of three test cases, there is a recall improvement compared to the Lexical Matcher. This growth is even more meaningful in the Anatomy test case, with a minimum increase of 14.5%. Despite that, there seems to be a large trade-off between recall and precision, bringing F-Measure lower than the Lexical Matcher. This precision loss may occur because although the word models are trained in corpora

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	**81.8%**	88.7%	97.9%	62.2%	76.0%
LM + WEM (PubMed)	74.8%	84.5%	79.3%	86.4%	80.9%	83.6%	70.8%	68.5%	69.7%
LM + WEM (PMC)	69.3%	84.2%	76.0%	83.9%	80.8%	82.3%	68.1%	68.3%	68.2%
LM + WEM (PMC + PubMed)	73.6%	**84.6%**	78.7%	85.3%	80.8%	83.0%	69.8%	68.4%	69.1%
LM + WEM (PMC + PubMed + Wikipedia)	70.9%	84.4%	77.0%	83.9%	80.9%	82.4%	68.6%	**68.6%**	68.6%

Table 5.3: Pre-trained word2vec models test and Lexical Matcher baseline. Word Embedding Matcher extends the Lexical Matcher and is filtered by a Strict Selector. All Matchers and Selector have a 0.6 threshold. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

inside the biomedical domain, these type of models have difficulties distinguishing homonyms, especially when using cosine similarity [1]. The recall between the different models is very similar, and there is no definitive model with better recall than the others. Nonetheless, precision fluctuates more between models, making the PubMed model the one with the highest precision in every test case, and consequently best F-Measure. This model requires less memory and time to load, hence, being the selected model to use in posterior evaluation.

The PubMed model was used to experiment with another approach, the extension of the Automatic Matching algorithm. By doing so, we could assess the potential growth of the mappings discovery, when taking into account the quality of the base alignment. This approach was tested using the same structure as in the Lexical Matcher extension, and its results are presented in Table 5.4.

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Automatic Matching	96.1%	83.6%	89.4%	96.8%	86.8%	91.5%	92.8%	74.0%	82.3%
AM + WEM (PubMed)	85.6%	**88.1%**	86.9%	91.1%	**87.3%**	89.2%	78.9%	**75.3%**	77.0%

Table 5.4: AML comparative strategies and Word Embedding Matcher extending Lexical Matcher and Automatic Matching without Background Ontologies, filtered by a Strict Selector. The Automatic Matching approaches use a 0.6 threshold while the Word Embedding Matcher and Selector use a 0.7 threshold. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

In this test, there is an increase in recall for all test cases, but with less relevance than in the Lexical Matcher extension. However, as in the previous results, there is a generalized precision reduction which

is very significant in the FMA-SNOMED test case with a decrease of 13.9%. These losses show that the word embedding models alone could be too general for the matching task, even when trained on the specific domain.

5.1.2 RDF2Vec Embedding Matcher

Opposite to word models which can be trained on a more generalizable corpus, RDF2Vec models embed graph entities and capture inherent properties only present inside that KG. Hence, the models require being trained on the very same KGs that are being tested. Also, RDF2Vec expects only one KG for training and without any established relations between two distinct KGs, similarities between their entities are practically non-existent. To mitigate this issue, we employed the Lexical Matcher to establish links between ontologies. Since Lexical Matcher achieves a generally high precision, we conjecture its mappings can be used to establish base connections, which allows RDF2Vec to traverse between ontologies during the random walk generation process over OWL *equivalent class* relations. To reduce training time and memory consumption, the only entities trained in the model were classes that belonged to a neighbourhood of a maximum of two degrees of separation from any of the Lexical Matcher alignment classes. In Figure 5.2 we present an example of a graph walk traversed in the described conditions.

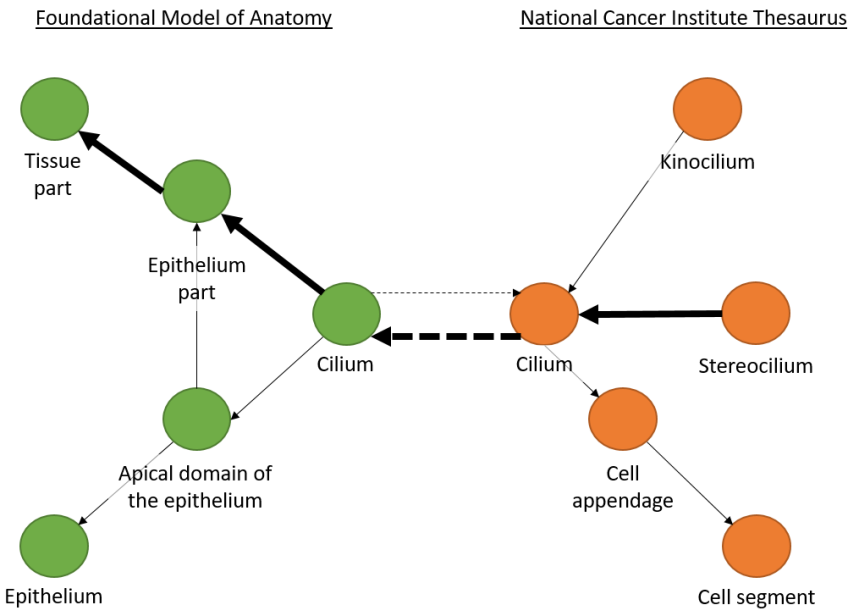

Figure 5.2: Representation of a graph walk with a path depth of 8 (both edges and nodes count for depth) between the FMA and NCI ontologies. The graph walk is portrayed by the largest bold arrows and the lexical matcher links by dashed arrows.

The RDF2Vec Embedding Matcher execution pipeline presented in Figure 5.3 can be described by

the following steps:

1. Running Lexical Matcher for the desired test case and outputting its alignment;
2. Creating a merged KG with *equivalent class* relations from Lexical Matcher alignment, resorting to a library named Owlready2 [32];
3. Using AML, computing the group of classes to train with a maximum of two degrees of separation from any of the Lexical Matcher alignment classes and writing them to a text file;
4. Training an RDF2vec model based on the merged KG and list of classes to train, using pyRDF2Vec library [60], and writing the model to a text file;
5. Executing a Java class to load the trained model and extend a certain alignment, similarly to the Word Embedding Matcher Java class, except that the cosine similarity is directly calculated inside the class.

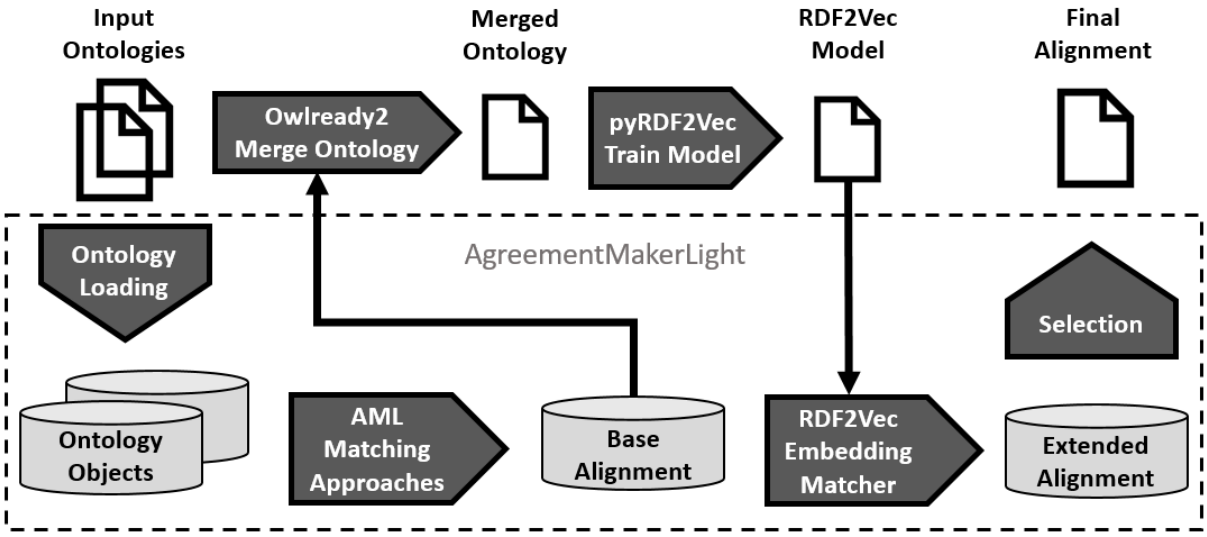

Figure 5.3: RDF2Vec Embedding Matcher Pipeline.

The pipeline was executed, training a model of random walks with a path depth of 8 and extending the Lexical Matcher. Given the obtained similarities were in general very elevated, the Matcher threshold was tested from 0.85 to 0.95 in intervals of 0.05. The produced results were unsatisfactory. Not only recall only increased by a maximum of 1,7% with a 0.85 threshold in one of the test cases, but precision declined notably. As the threshold increased, results got closer and closer to the original Lexical Matcher. This led us to a more detailed analysis of the data that was training the RDF2Vec model.

By nature, RDF2Vec explores all RDF relations solely on the directed path of the graph. It also includes the relations in the generated walks, which means that only paths leading to class ancestors were taken into account during embeddings training. Besides that, there is plenty of additional information like object and data properties that could be irrelevant information in the graph structure and diminish the

quality of the model. Therefore, diverse strategies of random walks generation were explored to attempt enhancing results. These strategies included extending RDF2Vec to enable walks generation on both directed or directed and reverse paths of the graph, exploring only relations of *subclass of* and *equivalent class*, including or excluding the relations URI from the paths (leaving only sequences of classes), and removing paths which contained the same class twice (meaning that the path was cyclic and traversed back to the same ontology due to *equivalent class* relations).

Nonetheless, the produced results of this study did not show significant changes. As information in paths decreased, precision declined considerably and recall increased slightly. Despite all the variations of the training data, the RDF2Vec strategy still obtained numerous false positives. This lead to investigating the possibility of filtering RDF2Vec discovered mappings before the selection process, according to a measure aside from graph embedding similarity. One intuitive way of filtering those mappings was to recompute the similarity of the mappings pairs with a AML Rematcher. The String Matcher, one of the Matchers that implement the Rematching feature inside AML, computes mapping similarity based on the ISub string metric [58] between the Lexicon names of both entities. We considered the String Matcher to be a beneficial tool for filtering RDF2Vec mappings without inserting refined matching algorithms in the process that call into question the reliability of the results.

The algorithm can be described as follows:

Algorithm 3: String Rematcher Algorithm

```
set thresh = matcher threshold;
set maps = the alignment with mappings to rematch;
set A = empty alignment;
foreach mapping in maps do
    set sourceClass = getSourceClass(mapping);
    set targetClass = getTargetClass(mapping);
    set sourceNames = names(sourceClass);
    set targetNames = names(targetClass);
    set maxSim = 0;
    foreach sourceName in sourceNames do
        set weight = weight(sourceName, sourceClass);
        foreach targetName in targetNames do
            set sim = weight * weight(targetName, targetClass);
            set sim = sim * stringSimilarity(sourceName, targetName);
            if sim > maxSim then
                set maxSim = sim;
            end
        end
    end
    if maxSim >= thresh then
        add (sourceClass, targetClass, maxSim) to A;
    end
end
```

With that in mind, we selected the walk generation alternative which had the highest recall, the directed path walks only with *subclass of* and *equivalent class* relations, filtered its extension with the String Rematcher and posteriorly filtered the remaining mappings using the Selector. Thresholds were obtained empirically to optimize F-measure. This approach obtained much more viable results.

In Table 5.5, we present the initial RDF2Vec test, the *subclass of* and *equivalent class* relations alternative (referred as REM* in this and upcoming Tables), and the final approach with the String Rematcher. In the initial strategy, there is a general slight increase in recall but a major decline in precision, especially in the Anatomy test case. The REM* strategy only managed to add a minimal recall boost in two of the three test cases, while decreasing precision drastically. In the Anatomy test case, precision dropped by 42.5% when compared with the Lexical Matcher. Even though, when coupled with String Rematcher

filter, the matcher achieved higher recall and F-Measure than in the Lexical Matcher in every test case. The String Rematcher succeeded not only in the filtering of false positives but also on the reassessment of mappings which RDF2Vec did not consider as important.

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	81.8%	88.7%	97.9%	62.2%	76.0%
LM + REM	79.0%	71.4%	75.0%	93.5%	82.4%	87.6%	84.6%	63.7%	72.7%
LM + REM*	53.6%	72.0%	61.4%	84.5%	82.5%	83.5%	78.9%	63.6%	70.4%
LM + REM * + SR	94.1%	**80.2%**	86.6%	95.8%	**85.4%**	90.3%	92.5%	**68.0%**	78.4%

Table 5.5: The three main RDF2Vec Embedding Matcher extension strategies, along with the Lexical Matcher Baseline. The Lexical Matcher uses a 0.6 threshold, the two first RDF2Vec strategies use a 0.85 threshold for the Matchers and Selector, and the last strategy uses a 0.7 threshold for the RDF2Vec Embedding Matcher and a 0.55 threshold for the String Rematcher and Selector. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

5.1.3 Embedding Mixture Matcher

This Matcher constitutes an experiment on combining graph similarities and word similarities as a single Secondary Matcher. In the graph component we would have the RDF2Vec Embedding Matcher and from the other, the Word Embedding Matcher and the String Matcher. To extend an alignment, after the exploration of the neighbourhood and creation of the candidate mappings list, both graph and word matchers are executed at the same time to calculate a linear combination of similarities from both word and graph components. Given a scalar a, the final similarity can be expressed by:

$$Sim = a * graphSim + (1 - a) * wordSim \tag{5.1}$$

Then, mappings are filtered according to a final threshold of the Embedding Mixture Matcher. This approach was tested for the two combinations of the graph and word components, a values in the interval [0.3, 0.8] with increments of 0.1, and final similarity values between [0.7, 0.9] with increments of 0.05. We used the PubMed model to feed the Word Embedding Matcher and the *SubClass* and *EquivalentClass* Paths model to supply the RDF2Vec Embedding Matcher. In Table 5.6, we present the results of the best parameter combination for each graph and word components mixture.

Both component mixtures managed to sustain proper precision while improving recall and F-Measure compared to the Lexical Matcher. The String Matcher with RDF2Vec Embedding Matcher combination was the most successful in both metrics. The word similarity seems to be most relevant than the graph similarity in the final decision, but oddly it was more significant in the less beneficial alternative. This demonstrates once again that the String Matcher and RDF2Vec Embedding Matcher are tools that go

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	81.8%	88.7%	97.9%	62.2%	76.0%
LM + EMM (SM + REM*) (a = 0.4, finalSim = 0.7)	95.0%	**76.8%**	85.0%	96.4%	**84.8%**	90.3%	95.0%	**67.3%**	78.8%
LM + EMM (WEM + REM*) (a = 0.3, finalSim = 0.85)	94.7%	74.7%	83.5%	95.0%	84.3%	89.3%	91.2%	66.9%	77.2%

Table 5.6: Embedding Mixture Matcher extending Lexical Matcher best results for the combination of the RDF2Vec Embedding Matcher with the String Matcher and the Word Embedding Matcher, along with the Lexical Matcher Baseline. The Selector threshold is equivalent to the final similarity threshold of the Embedding Mixture Matcher. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

along nicely. Besides, this type of strategy does not require training models with data outside the scope of the ontologies to match. Despite not achieving the best results, the union of the Word Embedding Matcher with the RDF2Vec Embedding Matcher was still advantageous. This unity managed to combine two tools which alone were still distant from outperforming the Lexical Matcher F-Measure, and together managed to surpass it in all test cases.

From all the Embeddings Similarity experiments, there are two primary strategies which are tied in terms of F-Measure, achieving the best performance in two of three test cases. One of them is the Lexical Matcher extended with RDF2Vec Embedding Matcher* and filtered with the String Rematcher. The other one is the Lexical Matcher extended with Embedding Mixture Matcher with String Matcher and RDF2Vec Embedding Matcher* as word and graph components respectively. Despite being tied in F-Measure, the first strategy achieved the best recall in all test cases, which meets our initial goal of discovering new mappings. Therefore, we consider this aspect determines it as the best plain embedding similarity strategy.

5.2 Embeddings Models

The previous experiments on plain embedding similarities only addressed unsupervised matching approaches. It is also possible to employ embedding based supervised matching approaches but requires circumventing the necessity of reference alignments to train supervised models, referred to in 1.2. To tackle this challenge, we developed a simple extension of an existing supervised matching method, Deep Align.

5.2.1 Unsupervised Deep Align

As previously mentioned, the Deep Align framework suffers from the lack of training data inside the biomedical domain. The framework uses supervised learning to retrofit word vectors to inscribe semantic similarity into sentence embeddings, and unsupervised learning to train the DAE to discover misalignments caused by embedding similarity based on relatedness instead of semantics. However, the only manner this system works is by training the framework on one of the biomedical benchmarks to apply the model on the others. One approach to solve this scalability problem is by producing training data which contained an acceptable small proportion of errors. Any other ontology matching system could produce training alignments. AML's results in OAEI biomedical benchmarks, especially in precision, make it a suitable candidate to create the required training data to feed Deep Align. This approach would define an unsupervised version of Deep Align in the sense that the framework does not require data labelled by human experts. In return, the alignment produced by Deep Align could be used to extend AML matching strategies.

To implement this approach, the first step was to transform all the matching required data for each test case into Deep Align's readable format. Namely, the entities main labels of both ontologies, reference alignments and word embeddings of the vocabulary present in the previous documents. Secondly, AML produced an alignment with one of its matching strategies and exported it to Deep Align's readable format as training data. Next, Deep Align's pipeline was executed, and the resulting alignment was exported. Lastly, AML imported the Deep Align's alignment, added the mappings to its own alignment, and applied a filter Selector.

This procedure was used to extend the Automatic Matching (without background knowledge) at first. Furthermore, we wanted to experiment with this method on the classical extension process of an alignment. Therefore, we also tested the procedure on extending and the Lexical Matcher alignment by one neighbourhood degree. This test was performed by writing into files the neighbour entities names from the alignment to extend, instead of the whole entity list inside the ontologies. Additionally, we experimented the Deep Align pipeline with and without the DAE to measure its influence on the extension process. The results of these tests are presented in Table 5.7. We also display the original Deep Align paper results separately in this table to analyze the contrast in performance. However, these results are not directly comparable since Deep Align uses a supervised learning model, trained over reference alignment mappings of a large test case.

The extension of both Automatic Matching and Lexical Matcher (with DAE) were able to increase recall and F-Measure in two of the three test cases. The recall improvement was even more significant on the extension of the Lexical Matcher, even though it only contained entities of one neighbourhood degree from the base alignment. The considerable decline of precision in the Lexical Matcher extension without the DAE show that this tool is very relevant in the discovery of false positives deriving from relatedness of names. The original paper results are still superior with a considerable margin, which we consider evident given the input this model was trained with. Using a reference alignment of a large and

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	81.8%	88.7%	97.9%	62.2%	76.0%
Automatic Matching	96.1%	83.6%	89.4%	96.8%	**86.8%**	91.5%	92.8%	74.0%	82.3%
AM + DA	94.4%	**86.6%**	90.3%	97.4%	83.7%	90.0%	90.8%	**75.8%**	82.7%
LM + DA (one neighbourhood degree)	97.6%	77.4%	86.3%	98.6%	79.5%	88.0%	97.0%	65.5%	78.2%
LM + DA (one neighbourhood degree, w/o DAE)	84.3%	81.6%	82.9%	85.6%	81.1%	83.3%	85.6%	66.8%	75.0%
Deep Align (Supervised)	96.8%	91.3%	94.0%	94.6%	89.2%	93.2%	93.1%	85.6%	89.2%

Table 5.7: Unsupervised Deep Align extension strategies, along with the Lexical Matcher, Automatic Matching, and the original Deep Align paper results. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

similar ontology matching test case is far more beneficial than training on an alignment of the same test case with potential errors. Therefore, our results are not comparable to such an ungeneralizable approach. Nevertheless, these experiments indicate that the Deep Align framework can be trained on data with a certain extent of errors and still provide value to other ontology matching strategies.

5.3 Comparison with State of the Art

Concluding the experiments of Embedding Similarities and Embedding Models, we present the best strategy of each approach in terms of F-Measure in Table 5.8. In two of the three test cases, the Embedding Model alternative obtained a considerable advantage in terms of recall and F-Measure. Nonetheless, in the FMA-NCI test case, the Embedding Similarity actually achieved the best performance in recall and F-Measure. Regardless, precision between the two alternatives does not differ significantly. This demonstrates that although tools like DAEs can be valuable, sometimes simpler alternatives can achieve similar results to more complex models.

In the following Tables 5.9, 5.10, and 5.11 we present our best strategies in Embedding Similarity and Embedding Models included in the performance tables of OAEI 2019 test cases of Anatomy, FMA-NCI small fragments and FMA-SNOMED small fragments respectively. Our strategies always achieve a position in the upper half of the tables. More in detail, a second and sixth places in Anatomy, a fourth and fifth places in FMA-NCI, and a second and fifth places in FMA-SNOMED. Our approach always achieves better results than other systems which do not use external ontologies/metathesauri as background knowl-

Strategy	Anatomy			FMA -> NCI			FMA -> SNOMED		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Lexical Matcher	96.1%	69.7%	80.8%	97.0%	81.8%	88.7%	97.9%	62.2%	76.0%
Automatic Matching	96.1%	83.6%	89.4%	96.8%	**86.8%**	91.5%	92.8%	74.0%	82.3%
LM + REM* + SR	94.1%	80.2%	86.6%	95.8%	85.4%	90.3%	92.5%	68.0%	78.4%
AM + DA	94.4%	**86.6%**	90.3%	97.4%	83.7%	90.0%	90.8%	**75.8%**	82.7%

Table 5.8: The best Embedding Similarity and Embedding Model strategies, along with all comparative strategies. The best recall in each test case is represented in bold while the best F-Measure is represented underlined.

edge but it also exceeds some of the systems which use them. This is particularly interesting for the LM + REM* + SR strategy because it uses no external models. Even though the approach requires time and computational resources to train an RDF2Vec model, it is more scalable than searching for appropriate external models or background knowledge that suit particular domains.

Matching System	Uses External Ontologies/ Metathesauri?	Anatomy		
		Precision	Recall	F-Measure
AML	Yes	95.0%	93.6%	94.3%
AM + DA	No	94.4%	86.6%	90.3%
LogMapBio	Yes	87.2%	92.5%	89.8%
POMAP++	Yes	91.9%	87.7%	89.7%
LogMap	Yes	91.8%	84.6%	88.0%
LM + REM* + SR	No	94.1%	80.2%	86.6%
SANOM	No	88.8%	84.4%	86.5%
Lily	Yes	87.3%	79.6%	83.3%
Wiktitionary	Yes	96.8%	73.0%	83.2%
LogMapLite	No	96.2%	72.8%	82.8%
ALIN	No	97.4%	69.8%	81.3%
FCAMap-KG	No	99.6%	63.1%	77.2%
StringEquiv	No	99.7%	62.2%	76.6%
DOME	No	99.6%	61.5%	76.0%
AGM	No	15.2%	19.5%	17.1%

Table 5.9: OAEI 2019 Anatomy test case participant systems performance sorted in descending order by F-Measure, along with the best Embedding Similarity and Embedding Models strategies highlighted in yellow.

Matching System	Uses External Ontologies/ Metathesauri?	FMA -> NCI		
		Precision	Recall	F-Measure
AML	Yes	95.8%	91.0%	93.3%
LogMap	Yes	94.4%	89.7%	92.0%
LogMapBio	Yes	91.9%	91.2%	91.5%
LM + REM* + SR	No	95.8%	85.4%	90.3%
AM + DA	No	97.4%	83.7%	90.0%
POMAP++	Yes	97.9%	81.4%	88.9%
LogMapLite	No	96.7%	81.9%	88.7%
FCAMapKG	No	96.7%	81.7%	88.6%
SANOM	No	97.9%	80.3%	88.2%
DOME	No	98.4%	76.6%	86.1%
Wiktionary	Yes	99.1%	60.8%	75.4%
AGM	No	49.5%	48.1%	48.8%

Table 5.10: OAEI 2019 FMA-NCI small fragments test case participant systems performance sorted in descending order by F-Measure, along with the best Embedding Similarity and Embedding Models strategies highlighted in yellow.

Matching System	Uses External Ontologies/ Metathesauri?	FMA -> SNOMED		
		Precision	Recall	F-Measure
AML	Yes	92.3%	76.2%	83.5%
AM + DA	No	90.8%	75.8%	82.7%
LogMapBio	Yes	93.1%	70.3%	80.1%
LogMap	Yes	94.7%	69.0%	79.8%
LM + REM* + SR	No	92.5%	68.0%	78.4%
AGM	No	46.3%	36.5%	40.8%
POMAP++	Yes	90.6%	26.0%	40.4%
FCAMapKG	No	97.3%	22.2%	36.2%
LogMapLite	No	96.8%	20.8%	34.2%
DOME	No	98.8%	19.8%	33.0%
Wiktionary	Yes	96.5%	17.0%	28.9%

Table 5.11: OAEI 2019 FMA-SNOMED small fragments test case participant systems performance sorted in descending order by F-Measure, along with the best Embedding Similarity and Embedding Models strategies highlighted in yellow.

Chapter 6

Conclusions and Future Work

In this dissertation, we developed and evaluated several embedding based ontology matching approaches to tackle the Biomedical KG matching task. Despite the development of sophisticated strategies in this area, systems' performance had plateaued for some years. The guiding hypothesis was that word embeddings based on external corpora and graph embeddings would be able to enrich KG matching approaches by accessing novel information (in the case of corpus-based embeddings) or difficult to process information (in the case of graph embeddings). In addition, this would also result in more generalizable KG matching approaches, independent of reference alignments to either train supervised models or guide the development of increasingly sophisticated approaches.

Experimented strategies ranged from simpler embedding similarities, based on word and graph embeddings, to more refined embedding models which required quality training data to perform with distinction. The obtained results revealed that although the developed strategies did not perform better than state of the art systems in existing benchmarks, the best strategies achieved very good performance. Moreover, the graph embeddings approach does not rely on external knowledge, and was able to perform better than any of the state of the art systems that also do not use external knowledge, and even some that do. This fulfills the goal of developing strategies that are able to generalize to tasks and domains where external knowledge is not available.

6.1 Future work

The most evident limitation of this work is the amount of resources needed to run the methodology. First of all, there is the training of the embedding models. The training process requires a large amount of time and memory. This was the principal motive to use pre-trained models when possible. Some of the RDF2Vec models took more than three hours to train only entities in the neighbourhood of two degrees from the base alignment, consuming more than 10 GBs of RAM. Training a graph embedding model with the union of two whole ontologies would not be possible without powerful computational resources.

Next, there is the matching process. Large biomedical ontology objects already occupy considerable memory space during the matching process. Loading additional embedding models to apply embedding strategies increases memory overhead. One of the small fragments task in the OAEI test cases, the SNOMED-NCI, could not run in any of the strategies due to the lack of memory. Additionally, the Deep Align framework pipeline employs a stable marriage algorithm which is rather inefficient *per se* and also requires storing a similarity matrix between every combination of entities to match, expending even more memory. Although the matching task is ideally supposed to be executed once per pair of ontologies to match, the matching systems should at least spend a reasonable limit of resources, according to the current costs of computational processing and memory. Developing more efficient algorithms to compute graph embeddings would be a relevant direction to explore.

Another limitation regards the use of word embedding models. Pre-trained word2vec models are based on single words, and their poor performance in the experiments may be due to the fact that a linear combination of word vectors is a too shallow representation of biomedical entities. As mentioned in 2.3.1, one of the directions to explore is to include TF-IDF weights in the calculation of word embedding cosine similarity. Even though AML excludes stop-words from the names of entities, words like *bone* or *cell* have a reasonable frequency in an ontology and don't portray the entity as much as other words. Another line to address is training textual embeddings models on more specific corpora, rather than an extensive collection of scientific articles, especially Paragraph Vector models which represent sentences in a more meaningful way.

This work has shown that Neural Embeddings are a valuable technique to address the challenge of biomedical KG matching. In particular, word embeddings approaches based on sophisticated models such as DeepAlign can be used successfully even in cases where a reference alignment is not available. Furthermore, graph embeddings approaches, independent of any external sources of knowledge, and combined with simple string matching, produce competitive results to the best state of the art systems, and are only surpassed by systems employing external knowledge. Potential avenues of future work include not only addressing the aforementioned limitations, but also evaluating the developed approaches in other domains.

References

[1] B. Beekhuizen, C. X. Cui, and S. Stevenson. Representing lexical ambiguity in prototype models of lexical semantics. In *CogSci*, pages 1376–1382, 2019. 32

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001. 1, 7, 8

[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global, 2011. 1, 7

[4] O. Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004. 27

[5] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013. 4, 14

[6] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30 (9):1616–1637, 2018. 13

[7] P. Cai, W. Li, Y. Feng, Y. Wang, and Y. Jia. Learning knowledge representation across knowledge graphs. In *AAAI Workshops*, 2017. 16

[8] S. Castano, A. Ferrara, S. Montanelli, and G. Varese. Ontology and instance matching. In *Knowledge-driven multimedia information extraction and ontology evolution*, pages 167–195. Springer, 2011. 11

[9] B. P. Chamberlain, J. Clough, and M. P. Deisenroth. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*, 2017. 12

[10] M. Cheatham and C. Pesquita. Semantic data integration. In *Handbook of big data technologies*, pages 263–305. Springer, 2017. 4

[11] G. O. Consortium. The gene ontology (go) database and informatics resource. *Nucleic acids research*, 32(suppl_1):D258–D261, 2004. 8

[12] I. F. Cruz, F. P. Antonelli, and C. Stroe. Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, 2(2):1586–1589, 2009. 19

[13] S. De Coronado, M. W. Haber, N. Sioutos, M. S. Tuttle, L. W. Wright, et al. Nci thesaurus: using science-based terminology to integrate cancer research results. In *Medinfo*, pages 33–37, 2004. 27

[14] G. K. de Vries. A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 606–621. Springer, 2013. 15

[15] K. Donnelly. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006. 3, 27

[16] L. Ehrlinger and W. Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4, 2016. 2

[17] J. Euzenat, P. Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007. 9, 10

[18] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 527–541. Springer, 2013. 19

[19] D. Faria, C. Pesquita, E. Santos, I. F. Cruz, and F. M. Couto. Agreementmakerlight: a scalable automated ontology matching system. *DILS 2014*, page 29, 2014. XV, 21

[20] D. Faria, C. Pesquita, I. Mott, C. Martins, F. M. Couto, and I. F. Cruz. Tackling the challenges of matching biomedical ontologies. *Journal of biomedical semantics*, 9(1):4, 2018. 3, 4, 20

[21] D. Faria, C. Pesquita, T. Tervo, F. M. Couto, and I. F. Cruz. Aml and amlc results for oaei 2019. *OM@ ISWC*, 2536:101–106, 2019. 21

[22] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996. 1

[23] L. Getoor and C. P. Diehl. Link mining: a survey. *Acm Sigkdd Explorations Newsletter*, 7(2):3–12, 2005. 15

[24] I. Harrow, E. Jiménez-Ruiz, A. Splendiani, M. Romacker, P. Woollard, S. Markel, Y. Alam-Faruque, M. Koch, J. Malone, and A. Waaler. Matching disease and phenotype ontologies in the ontology alignment evaluation initiative. *Journal of biomedical semantics*, 8(1):55, 2017. 4

[25] T. F. Hayamizu, M. Mangan, J. P. Corradi, J. A. Kadin, and M. Ringwald. The adult mouse anatomical dictionary: a tool for annotating and integrating data. *Genome biology*, 6(3):1–8, 2005. 27

[26] S. Hertling and H. Paulheim. Webisalod: providing hypernymy relations extracted from the web as linked open data. In *International Semantic Web Conference*, pages 111–119. Springer, 2017. 16

[27] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020. 2

[28] E. Jiménez-Ruiz, B. C. Grau, I. Horrocks, and R. Berlanga. Logic-based assessment of the compatibility of umls ontology sources. *Journal of biomedical semantics*, 2(S1):S2, 2011. 27

[29] E. Jiménez-Ruiz, A. Agibetov, M. Samwald, and V. Cross. We divide, you conquer: from large-scale ontology alignment to manageable subtasks. *Ontology Matching*, page 13, 2018. 16

[30] T. Kenter, A. Borisov, and M. De Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016. 15

[31] P. Kolyvakis, A. Kalousis, B. Smith, and D. Kiritsis. Biomedical ontology alignment: an approach based on representation learning. *Journal of biomedical semantics*, 9(1):21, 2018. XV, 10, 15, 21

[32] J.-B. Lamy. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017. 34

[33] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014. 13

[34] B. Li and L. Han. Distance weighted cosine similarity measure for text classification. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 611–618. Springer, 2013. 12

[35] A. Liang and M. Sini. Mapping agrovoc and the chinese agricultural thesaurus: Definitions, tools, procedures. *The New Review of Hypermedia and Multimedia*, 12:51–62, 06 2006. doi: 10.1080/13614560600774396. 3

[36] L. Ma and Y. Zhang. Using word2vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2895–2897. IEEE, 2015. 13

[37] C. Martins, D. Faria, and C. Pesquita. Visualization and editing of biomedical ontology alignments in agreementmakerlight. In *ICBO*, 2015. 21

[38] B. McBride. The resource description framework (rdf) and its vocabulary description language rdfs. In *Handbook on ontologies*, pages 51–65. Springer, 2004. 7

[39] D. L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004. 8

[40] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. XV, 12, 13

[41] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013. 13

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. XV, 4, 14

[43] S. Moen and T. S. S. Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, pages 39–44, 2013. 30

[44] C. J. Mungall, C. Torniai, G. V. Gkoutos, S. E. Lewis, and M. A. Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome biology*, 13(1):R5, 2012. 25

[45] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, 37(suppl_2):W170–W173, 2009. 3

[46] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017. 8

[47] C. Pesquita, D. Faria, E. Santos, and F. M. Couto. To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In *Proc. 8th ISWC ontology matching workshop (OM), Sydney (AU), page this volume*, 2013. 3

[48] J. Portisch and H. Paulheim. Alod2vec matcher. *OM@ ISWC*, 2288:132–137, 2018. 16

[49] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. 30

[50] P. Ristoski and H. Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016. 15

[51] C. Rosse and J. L. Mejino Jr. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of biomedical informatics*, 36(6):478–500, 2003. 27

[52] E. Santos, D. Faria, C. Pesquita, and F. M. Couto. Ontology alignment repair through modularization and confidence-based heuristics. *PloS one*, 10(12):e0144807, 2015. 20

[53] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer, 2014. 2

[54] H. Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007. 13

[55] P. Shvaiko and J. Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering*, 25(1):158–176, 2011. 9

[56] S. K. Sienčnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 239–243, 2015. 13

[57] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972. 13

[58] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *International Semantic Web Conference*, pages 624–637. Springer, 2005. 35

[59] W. R. Van Hage. Evaluating ontology-alignment techniques. *Retrieved August*, 12:2019, 2008. 3

[60] G. Vandewiele, B. Steenwinckel, M. Weyns, P. Bonte, F. Ongenae, and F. D. Turck. pyrdf2vec: A python library for rdf2vec, 2020. https://github.com/IBCNServices/pyRDF2Vec. 34

[61] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008. 15

[62] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990. 16